MASTER EN SCIENCES INFORMATIQUES

Amélioration de la qualité des applications Web une approche par heuristique

Boclinville, Samuel

Award date:
2009

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame De la Paix, Namur
Faculté d'informatique
Année académique 2008-2009

Amélioration de la qualité des applications Web : une approche par heuristique

Samuel Boclinville

Mémoire présenté en vue de l'obtention du grade de master en informatique.

Résumé

La qualité d'une application Web est un élément critique pour un développeur étant donné que celle-ci influence l'expérience de l'utilisateur et son comportement lors de la navigation. Les problèmes de qualité doivent donc être identifiés et corrigés en tenant compte des ressources disponibles. Nous présentons dans ce mémoire une approche générale qui utilise un modèle de qualité comme guide pour suggérer des recommandations permettant d'améliorer le niveau de qualité d'une application Web. Le processus de suggestion est réalisé par un algorithme de recherche heuristique. Ce dernier utilise un modèle probabiliste et une estimation de l'effort d'implémentation des changements pour suggérer la meilleure séquence de modifications à apporter pour améliorer la qualité. Notre approche, implantée à l'aide d'un algorithme du recuit simulé, a été testée sur un échantillon de 15 pages Web en utilisant un modèle de la qualité de navigabilité. Pour toutes les pages, l'algorithme heuristique recommande en temps réel (de l'ordre de la seconde) les meilleurs changements possibles. Par comparaison, une recherche exhaustive donne les mêmes résultats, mais avec un temps d'exécution exponentiel par rapport au nombre de changements possibles.

Mots-clés

Recommandation, Modèle de Qualité, Modèle Probabiliste, Application Web, Réseau Bayésien, Recuit Simulé, Coût d'Implémentation

Abstract

The quality of Web applications is a critical point for a developer since it influences the user experience and its behavior during the navigation. Quality problems must be identified and corrected given available resources. We present in this Master thesis a general approach that uses a quality model to suggest recommendations that improve the level of quality of a Web application. The suggestion process is done by a meta-heuristic algorithm which uses a probabilistic model and an estimation of the implementation cost to suggest the best sequence of changes to apply in order to improve the quality. Our approach, implemented with the simulated annealing algorithm, was tested on 15 different Web pages using a navigability model. For every page, the meta-heuristic correctly recommend the best possible changes in about a second. In comparison, an exhaustive search gives the same results with an execution time exponential to the number of possible changes.

Keywords

Recommendation, Quality Model, Probabilistic Model, Web Application, Bayesian Network, Simulated annealing, Implementation Cost

Remerciements

J'aimerais remercier mon promoteur Naji Habra pour son appui et son aide dans la structure et la rédaction de ce mémoire.

Mes remerciements vont pareillement à mon maître de stage Houari Sahraoui pour sa disponibilité et ses conseils très judicieux qui ont permis la réalisation de ce travail. Je le remercie également pour l'agréable expérience que j'ai eue en compagnie de son équipe, à l'Université de Montréal pendant quatre mois. J'en profite pour remercier tout particulièrement Stephane Vaucher pour ces nombreux conseils lors de mon stage à Montréal.

Je tiens particulièrement à remercier Stephane Vaucher, Houari Sahraoui, Naji Habra pour la publication d'un article scientifique accepté à la 9ème conférence internationale sur l'ingénierie web (ICWE2009) dont le sujet est directement relié au mémoire.

Je remercie également ma famille et mes amis pour leur aide dans la correction de ce mémoire.

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction

1.1 Contexte

L'avènement du Web a bouleversé la manière de consulter l'information, parfois au détriment du livre et autres écrits. Grâce à lui, de nombreuses personnes possédant un ordinateur avec une connexion internet ont un accès facile et rapide à une énorme quantité d'informations. Initialement, le Web 1.0 consistait en des pages web statiques « rarement » mises à jour. Cette conception initiale du Web a subi une première évolution transformant les sites statiques en sites Web dynamiques ou applications Web (parfois appelé Web 1.5) qui permet l'ajout d'éléments dynamiques à l'interface utilisateur. Ce changement a notamment été permis grâce aux bases de données stockant les informations contenues sur une page ainsi qu'aux langages de scripts. L'arrivée du Web 2.0 ou Web communautaire a changé la manière d'interagir avec les utilisateurs. Les internautes contribuent à alimenter les contenus des sites. Ils sont donc acteurs partageant et/ou créant de l'information. Les meilleurs exemples sont les blogs, wiki, etc. comme Ebay, Facebook, pour ne citer que ceux-là.

Chaque jour, de nombreux surfeurs potentiels, désirant partager et consulter de l'information, utilisent le Web. Vu le nombre important d'applications Web disponibles sur la toile, un surfer ne se contente pas seulement de choisir les applications adaptées à ses besoins mais privilégie celles qui permettent une expérience utilisateur plaisante. Une application Web réussie doit donc satisfaire à ces deux conditions de satisfaction des besoins des utilisateurs (critères de qualité fonctionnels) et d'expérience plaisante (critères de qualité non-fonctionnels). Pour une entreprise, un site Web peut être considéré comme un atout primordial aujourd'hui. Ce dernier lui fournissant une meilleure visibilité et un contact direct avec les consommateurs potentiels. La bonne qualité des applications Web est donc un facteur important de succès pour une entreprise.

L'évolution du Web pose de nouveaux problèmes de qualité. En plus d'un contenu statique et particulièrement pauvre, les structures et la topologie des sites Web 1.0 étaient relativement simples et maintenues manuellement. Aujourd'hui, le Web offre de nouvelles possibilités qui ont augmenté significativement la complexité des sites mais ont permis de satisfaire de nouveaux besoins. Les contenus des applications Web sont, par ailleurs, beaucoup plus riches que dans la première version du Web. Ces évolutions dues à une croissance rapide ont apporté des nouveaux problèmes de qualité, notamment du point de vue du management et de la maintenance des sites. Une application Web de bonne qualité permet dès lors une meilleure maîtrise des coûts et une maintenance plus faible.

La construction d'une application Web doit répondre à certains standards et règles qui vont permettre aux navigateurs d'afficher correctement la page créée. Le respect de ceux-ci, notamment définis par des organisations telles que le W3C[1], ne suffit pas pour garantir la bonne qualité d'une application. De plus, les sites Web évoluent, leurs complexités augmentent et les développeurs responsables de l'application ne sont pas toujours conscients des problèmes de qualité. En effet, il est nécessaire de pouvoir les identifier avant d'envisager une correction. Ceci justifie les nombreuses recherches réalisées dans le domaine de la qualité pour permettre d'évaluer et d'améliorer celle-ci.

1.2 Objectif et contribution

L'objectif de ce mémoire est de proposer des suggestions de modifications à implémenter sur une application Web dans le but d'améliorer son niveau de qualité. À partir d'un modèle de qualité, d'un ensemble de transformations possibles, et d'une estimation de l'effort d'implémentation, notre approche propose une méthode permettant de sélectionner la séquence optimale de transformations à appliquer sur l'application pour un effort limité. Trouver cette séquence est généralement insolvable dans un temps polynomial. Notre approche est donc basée sur un algorithme heuristique permettant d'approcher la meilleure solution.

Nous avons implémenté cette approche de recommandation et testé celle-ci sur 15 pages Web différentes sur lesquelles nous avons cherché à améliorer leurs navigabilités. L'algorithme heuristique du recuit simulé a permis la sélection des séquences de transformations. Nous montrons que notre approche identifie la meilleure séquence de transformations dans un temps raisonnable par rapport aux nombres de transformations disponibles, ce qui prouve que celle-ci est valable pour ce type de problème.

1.3 Organisation du mémoire

Ce mémoire est organisé comme suit. Le deuxième chapitre présente un état de l'art dans le domaine de l'évaluation et l'amélioration de la qualité des applications. Ce chapitre permet de résumer des travaux clés réalisés dans ces domaines et en fait ressortir des manques, par ailleurs, un faible nombre de contributions liant explicitement l'évaluation et l'amélioration de la qualité d'une application Web, une absence totale de prise en compte de l'effort d'implémentation dans une démarche d'amélioration de la qualité.

Le troisième chapitre traite les concepts nécessaires à la compréhension de ce mémoire. Il introduit une méthodologie de construction d'un modèle de qualité probabiliste qui utilise les réseaux bayésiens et en rappelle les bases nécessaires pour comprendre le fonctionnement de ce modèle. Ensuite, il présente un algorithme heuristique, le recuit simulé, guidant l'exploration d'un grand espace de solution dans un temps raisonnable.

Le quatrième chapitre explique notre approche de recommandation qui sélectionne la meilleure séquence de transformations à implémenter pour améliorer le niveau qualité d'une application Web pour un effort limité. Celle-ci utilise un modèle de qualité probabiliste, un ensemble de transformations candidates et une estimation de l'effort d'implémentation et consiste en deux processus : un processus d'évaluation pour évaluer les effets des transformations sur la qualité de l'application et un processus de recommandations qui utilise un algorithme heuristique pour proposer la séquence de transformations optimales pour améliorer la qualité.

Le cinquième et le sixième chapitre illustrent notre approche de recommandations sur une caractéristique de la qualité, la navigabilité. Le premier présente le modèle de qualité bayésien utilisé pour l'évaluation de la navigabilité et le second illustre le processus de recommandations qui utilise ce modèle de qualité et l'algorithme du recuit simulé.

Le septième chapitre traite de la réalisation de notre approche et montre sa faisabilité sur un cas pratique. Il détaille l'implémentation d'un outil permettant d'évaluer et de suggérer des recommandations pour améliorer la navigabilité d'une page et interprète ensuite les résultats d'une expérimentation réalisée sur 15 pages Web avec cet outil.

Enfin, le huitième chapitre conclut ce mémoire. Il présente une synthèse de ce travail et discute des limites et travaux futurs.

Chapitre 2

Etat de l'art des travaux réalisés dans le domaine de la qualité des applications Web

2.1 Introduction

Dans le domaine de la qualité du Web, différentes contributions ont proposé des principes, recommandations, suggestions pour *améliorer* la qualité des applications Web. D'autres ont suggéré des modèles, méthodologies et outils pour *évaluer* cette qualité. Certains travaux ont également été réalisés dans le domaine du refactoring, le but étant de modifier l'application pour en *améliorer* sa qualité. Ce chapitre présente quelques généralités sur les applications Web ainsi que certaines de ces contributions.

2.2 Les applications Web

Les applications Web, appelées également sites Web *dynamiques*, sont des logiciels applicatifs accessibles depuis Internet (ou Intranet) se manipulant à travers un navigateur Web.

Contrairement, aux sites Web *statiques* où les pages sont préalablement enregistrées, le contenu des pages Web d'une application est créé dynamiquement lorsque le navigateur envoie des requêtes à un serveur hébergeant le site Web. Le serveur utilise des technologies dynamiques du Web (ASP, JSP, PHP, ...) et des serveurs de bases de données pour générer le contenu et les navigateurs utilisent des langages de scripts, comme le javascript, pour ajouter des éléments dynamiquement à l'interface utilisateur.

L'avantage des sites Web et plus particulièrement des applications Web,

par rapport à une application logicielle, est son contenu maintenu et mis à jour, sans nécessité de déploiement ni d'installation de logiciel. Il suffit en effet d'un navigateur et d'une connexion Internet.

2.3 Principes et recommandations

Certains principes de recommandations et suggestions ont été proposés par des organisations telles que le W3C [27], IEEE [18] pour aider les développeurs à améliorer la qualité des applications Web. Des auteurs comme Nielsen et Loranger donnent des conseils pour faire face aux problèmes d'utilisabilité du Web [21]. De nombreuses autres contributions concentrent principalement leurs efforts sur une caractéristique de la qualité, l'utilisabilité [19].

D'autres auteurs ont réalisé des travaux sur la qualité en général des applications Web. Boldyreff & al présentent un ensemble de métriques pour mesurer le succès d'un site [7] et s'intéressent également à l'évolution de ceux-ci [4]. Dalton [10] a défini des critères permettant d'évaluer les applications en particulier leurs fonctionnalités, maintenabilités et utilisabilités. Deluze [11] a réalisé des études de performance des applications Web en analysant les connexions réseaux et la capacité des serveurs.

Ces contributions se contentent principalement d'énoncer des principes mais n'explicitent pas comment les utiliser pour évaluer la qualité d'une application Web.

2.4 Méthodologies d'évaluation de la qualité

Des auteurs ont proposé des méthodes d'évaluation de la qualité des applications Web. Olsina & al. [22] développent la méthodologie WebQEM (Web Quality Evaluation Methodology) pour évaluer la qualité d'une application pendant ses différentes phases du cycle de vie. Albuquerque & al. [3] présentent une méthodologie d'évaluation qualitative tenant compte des problèmes d'incertitude et d'imprécision des critères de qualités en utilisant le modèle flou FMSQE (Fuzzy Model for Software Quality Evaluation). Ces deux contributions utilisent des modèles de qualité définis sous forme d'arbre hiérarchique inspirés de la norme de qualité du logiciel ISO 9126 (détaillé plus loin) et des modèles classiques de la qualité. Ces modèles permettent d'estimer le niveau de qualité des applications Web en agrégeant l'évaluation de caractéristiques plus concrètes et donc mesurables.

Ces deux travaux sont directement reliés au travail de Malak et al. [19] qui propose une méthode pour construire un modèle de qualité évaluant les applications Web en utilisant les réseaux Bayésiens. Ce travail propose une méthode

d'évaluation qui se veut combler les problèmes d'incertitude, de précision et de subjectivité liés à l'évaluation de la qualité.

Caro & al. [8] présentent un outil d'évaluation de la qualité des données des protails Web se basant sur l'approche de Malek & al. où ils utilisent un réseau bayésien pour évaluer la qualité des données sur les portails Web.

2.5 Outils d'évaluation de la qualité

De nombreux travaux ont développé des outils dans le but d'évaluer quantitativement la qualité des applications Web. Parmi ceux-ci, on peut citer Brajnik [6] qui synthétise les outils permettant une évaluation automatique de la qualité des pages Web et de leurs interfaces graphiques. Dans cette contribution, il avance qu'il est possible d'automatiser, ou du moins en partie, l'évaluation de l'utilisabilité des sites Web par l'intermédiaire d'un outil. Cette contribution montre donc la faisabilité d'une approche automatique d'évaluation de la qualité des applications Web.

2.6 Méthodes de refactoring

Les travaux cités jusqu'à présent énoncent des principes ou se concentrent sur l'évaluation de la qualité. Aucun ne propose des méthodes permettant de modifier les applications Web dans le but d'améliorer leurs qualités. Ils se contentent principalement d'indiquer les critères responsables de la mauvaise qualité d'une application.

Par ailleurs, certains travaux de refactoring ont été réalisés dans ce but d'amélioration en appliquant des transformations sur le code source d'une application sans changer son comportement. Garrido & al. [16] montrent comment des patrons de refactoring (des transformations applicables sur une application Web) peuvent être appliqués aux modèles de navigation et de design dans le but d'améliorer la qualité. Ping & al. [24] proposent une approche permettant de transformer l'architecture d'une application Web en une architecture centré-controller.

Ces contributions reposent cependant sur un lien implicite entre qualité et refactoring. En effet, l'amélioration de la qualité est supposée garantie par le respect de bonnes pratiques sans se reposer sur une notion de qualité bien définie comme le fourni un modèle de qualité.

Olsina & al. [23] présentent un modèle Web de réfactoring capable d'améliorer de façon incrémentale deux critères de qualité externe (navigabilité et présentation) d'une application Web du point de vue de l'utilisateur final. Pour ce faire, les auteurs définissent des transformations sur le modèle de navigabilité et de présentation. Pour pallier le manque de lien explicite entre évaluation

et amélioration, cette contribution montre comment incorporer l'évaluation de la qualité dans le processus de refactoring en utilisant la méthodologie WebQEM (présenté ci-dessus). Les auteurs illustrent, sur un cas pratique, leur approche de refactoring Web pour améliorer l'utilisabilité d'une application Web où la WebQEM est utilisée pour évaluer les impacts du refactoring sur les attributs qualités.

2.7 Conclusion

Les travaux réalisés dans le domaine de la qualité du Web mettent en évidence un nombre conséquent de contributions réalisées dans le domaine de l'évaluation. Bien que ceux-ci se situent dans une démarche d'amélioration de la qualité, ils n'indiquent pas qu'elles sont les modifications à apporter pour pallier à ces problèmes de qualité.

Les méthodes de refactoring présentées montrent qu'il existe également des travaux ayant pour objectif d'améliorer la qualité d'une application Web. Cependant, la plupart des travaux réalisés ne comble pas le manque de lien entre une démarche d'évaluation et d'amélioration. En effet, certaines méthodes de refactoring, bien qu'elles améliorent la qualité d'une application, se basent sur de bonnes pratiques plutôt que sur une notion de qualité bien définie. Elles ne permettent donc pas d'évaluer la conformité des modifications apportées sur une application par rapport à un ensemble de règles/standards. Pour pallier ce manque, Olsina & al. [23] proposent une approche qui utilise un modèle de qualité pour évaluer les impacts des transformations sur la qualité.

Des travaux ont également été réalisés dans le domaine de la qualité du logiciel, plus particulièrement dans le domaine orienté-objet (OO), pour combler les manques entre l'évaluation et l'amélioration de la qualité. Par exemple, Sahraoui & al. [25] examinent la possibilité d'utiliser les métriques OO dans des situations de détections automatiques où des transformations peuvent être appliquées pour améliorer la qualité d'un système. Ce processus de détection utilise également un modèle d'estimation de la qualité pour analyser l'impact de transformations sur des métriques OO.

Ces travaux de réfactoring montrent que l'amélioration de la qualité, en particulier d'une application Web, est possible par l'intermédiaire de transformations applicables sur celle-ci. Cependant modifier une application demande un effort d'implémentation. Mais, à notre connaissance, aucune des contributions sur le refactoring Web ne prend en compte les ressources nécessaires à l'implémentation des transformations.

L'évaluation de la qualité d'une application Web doit pouvoir être réalisée

automatiquement. Cet automatisme est atteignable par l'intermédiaire d'un outil permettant d'évaluer quantitativement la qualité à partir d'attributs mesurables. Brajnik [6] montre cette possibilité d'évaluation automatique, ou dumoins partiellement automatique, pour l'utilisabilité.

Ce mémoire a pour objectif de présenter une approche outillée permettant d'améliorer automatiquement, ou du moins le plus automatiquement possible, la qualité des applications Web en prenant en compte l'effort d'implémentation et en combinant explicitement le processus d'évaluation et d'amélioration de la qualité. Cette approche se veut donc combler les manques dans le domaine de la qualité du Web.

Chapitre 3

Contexte vers l'élaboration d'une approche générale améliorant la qualité d'une application Web

3.1 Introduction

Les travaux présentés dans le chapitre 2 ont permis d'identifier des manques dans le domaine de la qualité du Web, et plus particulièrement dans le domaine de l'amélioration de la qualité des applications Web. Par exemple, peu de travaux permettent d'évaluer l'impact des améliorations sur la qualité d'une application. En effet, seuls Olsina & al. [23] illustrent cette possibilité en évaluant les effets des transformations apportées sur une application Web par l'intermédiaire d'un modèle de qualité.

L'objectif de ce mémoire est de définir une approche générale permettant de suggérer des recommandations, à partir de critères bien définis, pour améliorer le niveau de qualité d'une application Web. Notre approche de recommandations reprend le principe présenté par Olsina & al. qui utilise un modèle de qualité pour évaluer ces suggestions. Ce chapitre a pour objectif de présenter les notions nécessaires pour comprendre le fonctionnement des modèles de qualité et plus particulièrement le modèle de qualité Web probabiliste que nous allons utiliser pour notre approche.

Certains contributions, notamment dans le domaine de la qualité du logiciel, utilisent des méthodes heuristiques pour améliorer la qualité. Par exemple, Seng & al. [26] utilisent ces méthodes pour sélectionner les transformations à appliquer sur les métriques de design orienté-object (OO) dans le but d'en

améliorer la qualité. Notre approche de recommandation utilise également ce principe. Ce chapitre explique également les méthodes heuristiques, en particulier l'algorithme du recuit simulé que nous avons utilisé pour suggérer des recommandations.

Ce chapitre a donc pour objectif de présenter les pré-requis nécessaires pour comprendre notre approche qui améliore la qualité des applications Web.

3.2 Modèle de qualité

Un modèle de qualité (MQ) Web doit donner une bonne indication de la qualité d'une application Web. Ces indications peuvent être regroupées en catégories (par exemple, bon versus mauvais) et/ou valeurs de prédiction (par exemple, un score) mais doivent être représentatives de la perception d'un expert de la qualité. Par exemple, l'application possède une bonne ou mauvaise utilisabilité.

Un MQ consiste en un ensemble de critères permettant de déterminer le niveau de qualité d'une application. Pour représenter la relation qui existe entre les critères, ceux-ci sont reliés entre eux. Une évaluation effective de la qualité d'une application est possible par l'intermédiaire de méthodes de mesure qui permettent d'évaluer la valeur de chacun des critères. Un MQ estime donc le niveau de qualité d'une application à partir de valeurs de métriques extraites de celle-ci.

La création d'un MQ Web s'inspire généralement de règles et standards existants. Ceux-ci peuvent être issus de la littérature sur qualité du logiciel ou bien de recommandations et principes qualités du Web. Un MQ est responsable de structurer, systématiser et éventuellement automatiser l'évaluation de la qualité.

Tout modèle peut être utilisé comme base dans une démarche d'amélioration de la qualité d'un produit ou processus. Cependant celui-ci ne permet pas, de lui même, de définir quels sont les changements effectifs qui doivent être implémentés pour améliorer la qualité mais se contente généralement d'identifier des problèmes issus d'un critère ou d'une caractéristique qualité.

Dans le domaine de la qualité du logiciel, de nombreux standards et normes ont été définis. Ceux-ci tentent de refléter la qualité intrinsèque d'un produit à partir d'un ensemble de caractéristiques fonctionnelles et/ou non-fonctionnelles (comme par exemple l'utilisabilité). Par exemple, la Norme ISO 9126 issue de l'Organisation Internationale de Normalisation (ISO) définit et décrit un MQ capable d'évaluer la conformité d'un produit logiciel développé par rapport à des exigences formulées. Pour ce faire, la norme définit un ensemble de caractéristiques qualités. Celles-ci sont ensuite détaillées en sous-caractéristiques

comme le montre le tableau 3.1.

Tableau 3.1 – Caractéristiques et sous-caractéristiques de la qualité selon la norme ISO 9126

Caractéristiques	**Sous-caractéristiques**
Fonctionnalité	Aptitude Exactitude Interopérabilité Sécurité Conformité
Fiabilité	Maturité Tolérance aux fautes Possibilité de récupération
Utilisabilité	Compréhensibilité Facilité d'apprentissage Facilité d'exploitation Attractivité
Rendement	Performance Ressource
Maintenabilité	Facilité d'analyse Facilité de modification Stabilité Testabilité
Portabilité	Facilité d'adaptation Facilité d'installation Conformité Interchangeabilité

De nombreux travaux réalisés sur la qualité du Web se sont inspirés de cette norme pour définir un ensemble de critères permettant d'évaluer la qualité des applications Web. En effet, comme présenté dans le chapitre 2, Malak [19], Olsina [22] et Albuquerque [3] ont repris les caractéristiques définies pour l'évaluation de la qualité d'un produit logiciel par l'ISO dans le cadre du Web.

Malak a également synthétisé les travaux réalisés dans le cadre de la qualité du Web, ce qui lui a permis de souligner les nombreuses divergences entre les auteurs. En effet, la plupart des auteurs des contributions ne s'entendent pas sur la définition ni sur la nature des critères. Chacun a sa propre interprétation de la qualité. Malak a donc envisagé l'évaluation de la qualité des applications Web de façon probabiliste en définissant une méthodologie de construction d'un MQ Web. Nous allons voir ce qui justifie une approche probabiliste.

3.3 Modèles de qualité probabiliste

Il existe de nombreux problèmes inhérents à l'évaluation de la qualité d'un produit logiciel, en particulier une application Web. En effet, la création d'un MQ reste un problème subjectif lors du choix des critères de qualité et un problème d'incertitude dû aux regroupements de ceux-ci. Par exemple, comment mesurer la fiabilité d'un système objectivement ? La notion de fiabilité est et reste un problème subjectif. Par conséquent, aucun MQ ne peut refléter une estimation objective de la qualité. Ceci se reflète dans la littérature. En effet, les auteurs de travaux sur la qualité ont leur propre interprétation de la notion de qualité [19].

Utiliser une modèle probabiliste permet de traiter ces problèmes d'incertitude et subjectivité par l'intermédiaire de probabilités. Celles-ci peuvent également être utilisées pour refléter l'imprécision liée à la mesure de la valeur d'un critère [19]. Des auteurs, tels que Malak, ont utilisé des MQ probabilistes pour pallier à ces différents problèmes.

Dans sa thèse, Malak propose une méthodologie de construction d'un MQ des applications Web en utilisant les réseaux bayésiens. Ceux-ci sont une bonne alternative pour raisonner dans l'incertitude et peuvent être utilisés pour juger si une application est de bonne ou de mauvaise qualité en indiquant dans quelle mesure cela reste valable. Sa contribution synthétise les facteurs de qualité spécifique aux applications Web.

Malak a implémenté un MQ bayésien permettant une estimation précise du niveau de navigabilité d'une page Web. Pour ce faire, elle a défini un ensemble de critères mesurables pour la navigabilité et a converti ces valeurs de métriques en probabilités pour qu'elles soient exploitables par le réseau bayésien, parfois en faisant appel à la logique floue. Grâce aux réseaux bayésiens, son modèle l'avantage d'être évolutif, extensible et adaptable.

Dans le cadre de cette thèse, nous utilisons le travail de Malak et plus particulièrement son réseau de navigabilité. La section suivante a pour objectif de présenter les notions de base nécessaire à la compréhension des réseaux Bayésiens et la logique floue.

3.3.1 Rappel sur les Réseaux Bayésiens

Le théorème de Bayes

Le théorème de Bayes est la base des réseaux Bayésiens (RBs). Ce théorème permet de déterminer la probabilité de A sachant B si la probabilité de A, B

et B sachant A sont connus :

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

La probabilité $P(A)$ (resp.$P(B)$) est appelée probabilité a priori de A (resp. de B). Cette probabilité à priori de A (resp. B) signifie qu'elle est « antérieure », c'est-à-dire qu'elle précède toute information sur B (resp. A). $P(B)$ est également appelée probabilité marginale. La probabilité $P(A|B)$ est appelée la probabilité à posteriori de A sachant B, ce qui signifie qu'elle est « postérieure » à B et en dépend directement. La probabilité $P(B|A)$ est appelée la fonction de vraisemblance de A.

Structure d'un réseau bayésien

Les RBs sont issus de la combinaison entre la théorie des graphes et la théorie des probabilités. Ce sont des graphes dirigés acycliques permettant de modéliser des relations causales. Les noeuds représentent des variables aléatoires et les arêtes orientées définissent les relations de cause à effet entre ces noeuds (de « parents » vers « fils »).

Une variable aléatoire est une fonction définie sur un ensemble d'événements/états possibles (par exemple : « Oui » et « Non ») lors d'une expérience aléatoire. Leur valeur représente la probabilité que l'événement se produise. La somme des valeurs des probabilités des états d'une variable aléatoire doit être égale à 1.

Pour les RBs, les noeuds parents affectant un même fils doivent être des variables indépendantes.

Un noeud sans aucun parents possède une table de probabilités « à priori ». Le noeud A de la figure 3.1 a une probabilité de 60% que la valeur « Oui » se produise et une probabilité de 40% pour la valeur « Non ».

Chacun des autres noeuds modélise une relation d'incertitude avec ses noeuds parents par l'intermédiaire d'une table de probabilités conditionnelles. Ces tables définissent la distribution de probabilité de chacun des noeuds conditionnellement à ses noeuds parents et servent à définir la relation entre les noeuds du réseau. Pour un noeud X avec 2 parents (A et B), la table de probabilités de X est réalisée sachant ses parents ($p(X|A, B)$). La figure 3.1 montre un exemple de table pour ce noeud X où la probabilité que l'événement X soit égale à « Oui » sachant que A et B sont « Oui » est de 50% et la probabilité que l'événement X soit également à « Oui » sachant que A est « Oui » et B est « Non » est de 90%.

En général, la détermination des valeurs pour ces tables sont faites à partir de données empiriques et jugement d'experts.

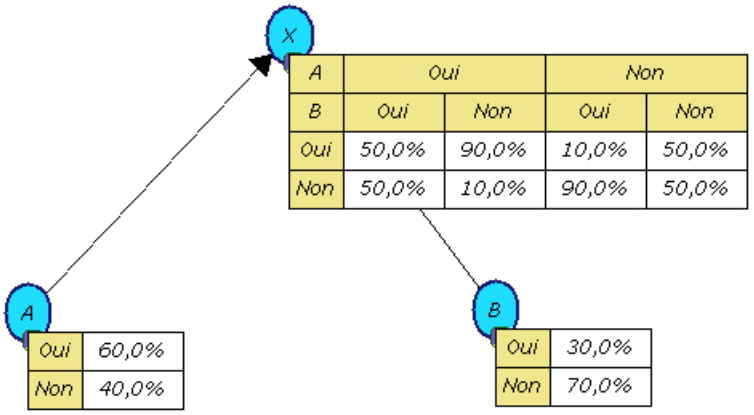


FIGURE 3.1 – Exemple de table de probabilités

Dans un réseau bayésien, les relations parents-fils entre les variables ne sont pas déterministes mais bien probabilistes. Donc, une observation d'une ou plusieurs causes (noeuds parents) n'entraîne pas systématiquement le ou les effets qui en dépendent (noeuds enfants). En fait, elle augmente ou diminue la probabilité de les observer. Les RBs permettent donc d'intégrer l'incertitude dans le raisonnement en tenant compte de la connaissance des experts (via le graphe) et de l'expérience dans les données (via les paramètres des noeuds).

Raisonner avec un réseau bayésien

Les RBs s'avèrent être efficaces pour raisonner dans l'incertitude. Ils sont, par exemple, utilisés dans des applications réelles de prédiction pour des systèmes critiques. Ils permettent également de supporter la prise de décision basée sur plusieurs critères.

Fenton et Neil ont réalisé de nombreuses contributions en exploitant les possibilités offertes des RBs. Les auteurs les utilisent pour aider à prendre des décisions dans le management du risque [12]. Les RBs servent également dans le domaine du génie logiciel comme moyen de prédictions des défauts et de fiabilité [13]. Une autre contribution [14] combine les RBs avec des techniques d'aide à la décision multi-critère pour décider des composants de sécurité. Ce travail illustre également les possibilités de raisonnement des RBs sur un problème de voyage, comme le montre la figure 3.2. Ces réseaux sont donc utilisés pour modéliser notamment l'heure de vol (depart time), les moyens de transport (transport type) possibles (taxi, voiture, train), l'heure de départ (start time) et le temps d'attente à l'aéroport (waiting time).

Les RBs sont des mécanismes puissants pour faire de l'inférence et donc raisonner. Résoudre un réseau revient à déterminer les probabilités condition-

FIGURE 3.2 – Exemple de modélisation d'un réseau bayésien pour un problème de voyage

nelles d'un noeud ou d'un ensemble de noeuds en fonction des instances des valeurs d'autres noeuds. Concrètement, la définition de factorisation permet de calculer la valeur d'un noeud A d'un réseau V sachant ses parents :

$$P(A) = \prod_{A \in V} (P(A|parents(A)) \tag{3.1}$$

où $parents(A)$ est l'ensemble de parents de A.

Les RBs offrent la possibilité d'instancier certaines valeurs de noeuds, les autres noeuds s'adaptent en fonction des valeurs fixées. Pour le problème de voyage de la figure 3.2, le réseau permet deux types de raisonnement : il est possible, d'une part, de faire de l'inférence vers l'avant en fixant le type de transport et l'heure de départ pour en connaître le temps d'attente (voir figure 3.3), ou bien, de faire de l'inférence vers l'arrière en déterminant le temps d'attente et les heures de départ dans le but de savoir quel type de transport est préférable (voir 3.4). Ce problème illustre bien les possibilités de raisonnement des RBs.

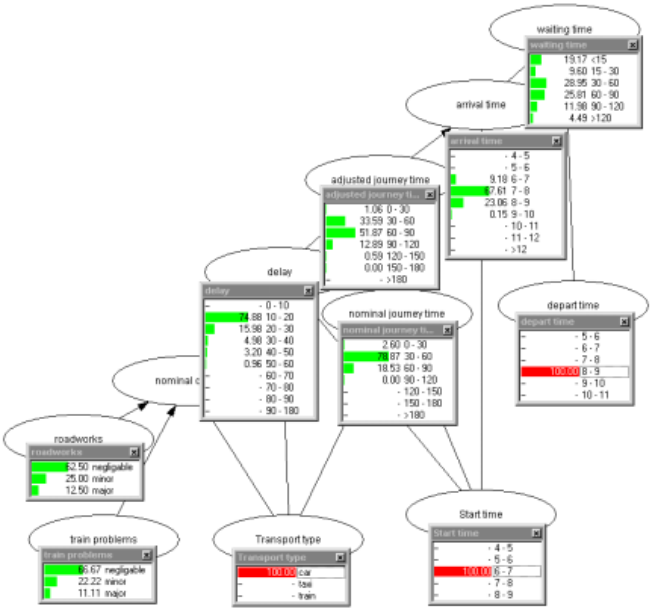


FIGURE 3.3 – Exemple de raisonnement (inférence avant) avec un réseau bayésien pour un problème de voyage

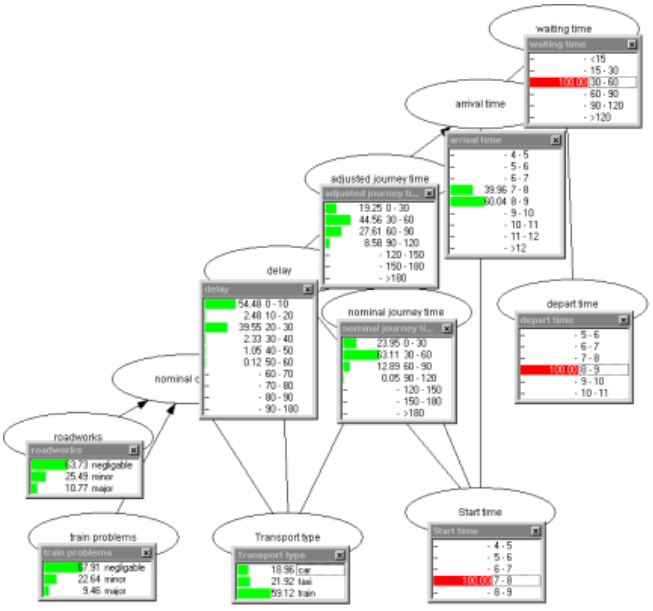


FIGURE 3.4 – Exemple de raisonnement (inférence arrière) avec un réseau bayésien pour un problème de voyage

Malak utilise les possibilités de raisonnement dans le cadre du Web où les RBs sont utilisés pour calculer une estimation de la qualité des applications Web à partir de différents critères mesurables.

La complexité des réseaux bayésiens

Résoudre un réseau bayésien est un problème NP-hard [9]. En effet, déterminer les probabilités conditionnelles d'un noeud est un problème de calcul coûteux. La complexité en temps est exponentielle à la densité du réseau et au nombre d'états des variables aléatoires de chaque noeud.

De nombreux algorithmes permettent de faire de l'inférence avec les RBs. Deux types d'approches sont possibles pour calculer la probabilité conditionnelle désirée : *l'inférence exacte* et *l'inférence approximative*. La première permet de calculer exactement la valeur d'un noeud mais nécessite un temps de calcul plus conséquent que la deuxième méthode qui l'approche. L'inférence exacte reste cependant valable pour des problèmes de petite taille. L'algorithme le plus connu pour ce type de calcul (inférence exacte) est l'algorithme Lauritzen-Spiegelhalter (LS) [20].

3.3.2 La logique floue

La logique floue étend de la théorie des ensembles classiques pour prendre en compte les ensembles définis de façon imprécise. Contrairement à la logique booléenne, cette logique permet d'autres états que Vrai et Faux.

Dans le cadre de l'évaluation de la qualité par un réseau bayésien, cette logique peut être utilisée pour convertir en probabilités certaines valeurs de métriques extraites d'une page Web. Ces probabilités sont alors utilisées par les noeuds d'entrée du réseau bayésien pour évaluer la qualité d'une application Web.

La théorie floue est cependant différente de la théorie des probabilités. En effet, la première décrit l'ambiguïté d'un événement en mesurant la fréquence de production de son occurrence alors que la seconde décrit l'incertitude de l'occurrence d'un événement. Mais, sous certaines conditions, la logique floue peut être utilisée comme probabilité.

En effet, Malak utilise la logique floue pour faciliter la conversion en probabilité des valeurs de métriques dont le nombre est potentiellement infini. Après une conversion de ces valeurs en variables discrètes ayant un nombre limité de valeurs, un processus de partitionnement flou remplace les différentes valeurs de métriques par un ensemble de fonctions qui représentent le degré d'appartenance de chaque valeur aux différents classes floues (souvent « Grand », « Moyen » et « Petit »)[19]. Les tables de probabilités de ces noeuds d'entrées

sont obtenues en calculant le pourcentage d'appartenance de la valeur de la métrique à la classe. Ce processus complet permettant la conversion de ces valeurs de métriques en probabilités est présenté en détail dans la thèse de Malak [19].

3.4 Méthodes heuristiques

3.4.1 Introduction

Tout problème de recherche implique un parcours de l'espace de recherche pour trouver la solution optimale. Ceci peut cependant demander un temps de calcul conséquent, si l'espace est particulièrement grand, voir infini. En effet, une façon naïve de procéder serait de tester l'ensemble des solutions possibles pour en sélectionner la meilleure. Cette approche n'est cependant pas praticable si le nombre de solution à envisager est particulièrement élevé. Car trouver la meilleure solution peut demander un temps de calcul conséquent. Dans ce cas, l'utilisation de méthodes heuristiques peut savérer une bonne alternative. Celles-ci guident l'exploration de l'espace de recherche pour trouver une solution proche de l'optimale dans un temps raisonnable [15].

Dans le domaine de la qualité du logiciel, l'objectif principal des méthodes de refactoring consiste à déterminer la séquence de transformations optimales. Dans le cas où le nombre de transformations est particulièrement grand, des méthodes heuristiques sont utilisées pour sélectionner les meilleures transformations à appliquer comme le montre Seng & al. [26] et Harman & al.[17] pour les programmes OO.

Notre approche de recommandation utilise également des méthodes heuristiques pour améliorer la qualité d'une application Web. Cette section a pour but d'expliciter qu'est-ce qu'un algorithme heuristique et dans quelles conditions celui-ci peut être utilisé. Nous terminons cette section par une présentation détaillée de l'algorithme heuristique du recuit simulé utilisé dans le cadre de ce mémoire.

3.4.2 Qu'est-ce qu'une recherche par méthode heuristique

Explorer un espace de recherche dans le but d'optimiser le résultat d'une ou plusieurs fonctions, appelé fonction objective, est un problème d'optimisation combinatoire. Cette recherche peut être réalisée par l'intermédiaire de différentes méthodes qui dépendent principalement de la taille de l'espace de recherche. Deux grands types de techniques sont utilisés pour résoudre ces problèmes d'optimisations : les méthodes exactes et les méthodes heuristiques. Tandis que les méthodes exactes trouvent dans tous les cas la meilleure solution, ces méthodes s'avèrent inefficaces pour un espace de recherche parti-

culièrement grand car celles-ci nécessitent de parcourir l'ensemble des solutions possibles pour en sélectionner l'optimale. La recherche d'une solution peut donc demander un temps de résolution non réaliste. Ce problème de performance peut être pallié grâce aux méthodes heuristiques qui permettent de chercher une solution dans un temps raisonnable. Cependant, ces méthodes ne garantissent pas de trouver la meilleure solution.

Les méthodes de recherche heuristiques sont implémentées par l'intermédiaire d'algorithmes heuristiques. Il existe deux grandes classes d'algorithmes : les algorithmes génétiques dont le principe est basé sur l'évolution d'une population de solutions et les algorithmes basés sur une recherche locale. Ces derniers comprennent notamment le recuit simulé et la recherche avec tabous. Dans le cadre de ce mémoire, nous allons utiliser l'algorithme du recuit simulé.

3.4.3 Un exemple d'algorithme heuristique : le recuit simulé

L'algorithme heuristique du recuit simulé s'inspire d'un processus thermique issu de la métallurgie qui vise à minimiser l'état d'énergie d'un solide. Ce processus tente de solidifier un matériau liquide en alternant des phases de refroidissement lent de la température et de réchauffement, appelé recuit [2]. L'algorithme du recuit simulé fait donc l'analogie entre la recherche d'une solution optimale pour les problèmes d'optimisation combinatoire et la recherche d'un état d'énergie solide minimal.

Le recuit simulé peut être utilisé pour guider l'exploration d'un large espace de recherche dans le but de trouver une solution proche de l'optimale. Cet algorithme se retrouve dans la littérature avec de légères variantes. Dans le cadre de ce mémoire, nous nous inspirons du livre « Simulated annealing and Boltzmann machines » [2] et de la thèse de Bouktif [5] pour en expliquer son principe et l'ajustement des différents paramètres.

Principe du recuit simulé

Par analogie au processus physique, l'algorithme du recuit simulé, représenté en pseudo code via l'algorithme 1, cherche une solution optimale minimisant l'état d'énergie en faisant décroître à chaque itération une température Trs (parfois appelée paramètre de contrôle) à partir d'une température initiale Trs_0. A partir d'une solution initiale s_0, l'algorithme explore d'autres solutions selon la procédure suivante :

À chaque itération, une nouvelle solution s' est générée par la fonction de transition V. Cette fonction est responsable de la création d'une solution dans le voisinage de s. Une fonction objective Obj compare ensuite ces deux solutions. Dans le cas où la solution voisine s' est meilleure, celle-ci remplace la solution courante pour la prochaine itération, sinon la solution courante

est possiblement remplacée avec une probabilité dépendant de la température courante. Cette acceptation d'une solution par probabilité permet d'éviter la stagnation de l'algorithme vers une solution locale et correspond à la phase du recuit dans la métallurgie. La probabilité d'acceptation d'une nouvelle solution dépend de la fonction exponentielle $e^{\Delta Obj/Trs}$. L'itération se termine en comparant la nouvelle solution avec la solution optimale provisoire $s_{Meilleure}$ et la remplace dans le cas où celle-ci est meilleure.

Après Nrs itération pour une même température, l'algorithme fait décroître la température géométriquement. Cette diminution de température est réalisée par la fonction $Trs := \alpha.Trs$. Finalement, l'algorithme s'arrête avec la solution $s_{Meilleure}$ lorsque la température (le paramètre de contrôle) s'approche de 0.

Comme toute méthode heuristique, l'algorithme du recuit simulé doit être adapté à chaque problème spécifique. Celui-ci (avec sa fonction objective $Obj(s)$ et sa fonction de transition V) ainsi que ses paramètres (la longueur de la température Nrs, la température initiale Trs_0, la décroissance de la température avec le paramètre α, le critère d'arrêt *StopCriteria*) ont besoin d'être défini en fonction d'un problème de recherche particulier.

Fonction objective

Tout problème d'optimisation combinatoire, nécessite une ou plusieurs fonctions objectives responsables d'évaluer la « valeur » d'une solution. Pour l'algorithme du recuit simulé, une seule fonction est nécessaire. Ce problème d'optimisation revient donc de minimiser ou maximiser la valeur de cette fonction objective. L'algorithme représenté en 1 cherche une solution optimale minimisant cette valeur comme le processus du recuit utilisé en métallurgie minimise l'état d'énergie. Pour un problème de maximisation, il suffit de changer les signes lors de l'acceptation d'une solution.

Fonction de transition

La fonction de transition est responsable de générer une solution voisine s' à partir d'une solution courante s ($s' \in V(s)$). Bien que cette fonction dépende du problème traité, il est préférable d'utiliser une fonction de transition qui ne modifie que légèrement la solution.

Longueur de la température

La longueur de la température (Nrs) représente le nombre d'itérations pour une même température. Plus sa valeur est grande, plus il est possible d'obtenir des solutions à partir d'une solution courante. Sa valeur, généralement déduite

```
RECUIT SIMULÉ (Trs_0, Nrs) ;
Choisis s_0                    /* Choix d'une solution initiale */ ;
Trs := Trs_0                 /* Initialisation de la température */ ;
STOP := false ;
s := s_0                /* Initialisation de la solution courante */ ;
s_Meilleure := s_0
                     /* Initialisation de la Meilleure Solution */ ;
while !STOP do
    for i := 1 to Nrs do
        Génère s' ∈ V(s)                  /* Génération d'un voisin */ ;
        ΔObj := Obj(s') − Obj(s)           /* Fonction objective */ ;
        if ΔObj ≤ 0 then
            s := s'                          /* Accepte la solution */ ;
        else
            Génère un nombre aléatoire r ∈ [0, 1] ;
            if r ≤ e^(−ΔObj/Trs) then
                s := s'
                      /* Accepte avec une petite probabilité */ ;
            end
        end
        if Obj(s') ≤ Obj(s_Meilleure) then
            s_Meilleure := s'          /* Meilleure solution trouvée */ ;
        end
    end
    Trs := α.Trs                          /* Diminue la température */ ;
    if StopCriteria then
        STOP := True ;
    end
end
return s_Meilleure ;
```

Algorithme 1: Algorithme du recuit simulé

après un certain nombre d'expériences réalisées au préalable, est fixée dans le but d'atteindre un certain équilibre.

Température initiale

La valeur initiale de la température (Trs_0) doit être suffisamment grande pour permettre virtuellement que toutes les transitions soient acceptées. Selon le livre ≪ Simulated annealing and Boltzmann machines ≫, ceci est possible si le taux d'acceptation initial ($e^{-\Delta Obj/Trs}$) est proche de 1.

Décroissance de la température

Dans l'algorithme heuristique présenté en 1, la fonction géométrique $Trs := \alpha.Trs$ permet de faire décroître progressivement la température. Bien qu'il existe d'autres fonctions dans la littérature, nous avons choisi d'utiliser la fonction géométrique où le paramètre α doit avoir une valeur proche de 1. Le livre ≪ Simulated annealing and Boltzmann machines ≫ conseille d'utiliser une valeur généralement comprise entre 0, 8 et 0, 99.

Critère d'arrêt

Le critère d'arrêt *StopCriteria* permet d'arrêter le processus de recherche d'une solution lorsque la température tend vers 0.

Chapitre 4

Méthodologie pour une approche générale de recommandations

4.1 Introduction

L'objectif de ce mémoire est de suggérer des recommandations pour améliorer le niveau de qualité d'une application Web donnée. Toute démarche d'amélioration de la qualité doit tenir compte des ressources disponibles et prendre en compte l'effort engendré par l'implémentation des suggestions par rapport au gain de qualité apporté. En effet, une recommandation doit être le résultat d'une analyse coût-bénéfice. La difficulté de cette approche revient à déterminer les modifications qui donnent le meilleur niveau de qualité d'une application Web pour un faible effort.

Ce mémoire s'inspire de principes utilisés dans le domaine de la qualité (du logiciel et du Web) et se veut donc combler les manques dans le domaine du Web énoncés dans la chapitre 2. Notre contribution est unique et consiste en les caractéristiques suivantes :

- Une approche générale d'évaluation et d'amélioration de la qualité d'une application Web
- Un lien explicite entre l'évaluation et l'amélioration de la qualité
- Une prise en compte de l'effort d'implémentation
- Une évaluation de la qualité par un modèle de qualité probabiliste
- Une utilisation d'un algorithme heuristique pour suggérer des recommandations

- Un outil capable d'évaluer la qualité et de suggérer des recommandations

Ce chapitre a pour objectif de présenter notre approche générale de recommandations reprenant les caractéristiques définies ci-dessus.

4.2 Approche de recherche

Notre approche de recommandations utilise des transformations pour améliorer le niveau de qualité d'une application Web considérée, comme les méthodes de refactoring, et un modèle de qualité (MQ) pour évaluer les effets de ces transformations sur la qualité de l'application. En effet, pour toute recommandation potentielle, il est nécessaire de pouvoir évaluer les modifications apportées selon une notion de qualité bien définie. Un MQ peut fournir cette évaluation en estimant précisément le niveau de qualité d'une application Web selon un ensemble de règles/standards. Comme présenté dans le chapitre 2, d'autres contributions utilisent ce principe de refactoring utilisant un MQ pour évaluer les impacts des transformations sur la qualité, telles que Olsina & al. [23] pour le Web et Sahraoui & al. [25] pour les logiciels OO.

Dans notre approche, une transformation est perçue comme une modification simple sur une application Web et correspond à une tâche bien définie d'un développeur. Le MQ calcule un score estimant le niveau de qualité, à partir de critères mesurés sur l'application.

Notre objectif est donc de combiner l'évaluation (par un MQ) et l'amélioration de la qualité (par des transformations) pour combler l'absence de lien explicite entre ces deux méthodes. En effet, les MQ ne permettent pas, par eux-mêmes, de déterminer quelles sont les meilleures modifications à apporter sur une application Web. Ils se contentent d'identifier les critères responsables de la mauvaise/bonne qualité de l'application. Et les effets des transformations doivent être évalués sur base d'une notion de qualité bien définie comme le font les MQ.

Deux types d'approches sont possibles pour améliorer la qualité à partir d'un MQ et de transformations. La première consiste à sélectionner et appliquer des transformations qui améliorent le niveau de qualité d'une application Web. Le MQ permet, dans ce cas, d'évaluer les changements de qualité engendrés par les transformations et de sélectionner les meilleures modifications (Approche « forward »). La seconde approche consiste à définir d'abord un seuil minimal de qualité à atteindre. Le MQ indique les critères à modifier pour obtenir ce niveau de qualité. Pour que ces changements correspondent à une tâche bien définie d'un développeur, il suffit ensuite de trouver des transformations qui modifient ces critères de qualité (Approche « backward »).

Dans les deux cas, une transformation modifie un ou éventuellement plusieurs critères de qualité. Toute transformation doit cependant rester cohérente par rapport à ce qu'elle modifie. Par exemple, si une transformation est responsable de l'ajout d'un moteur de recherche, celle-ci ajoute également tous les éléments qui composent un moteur de recherche. Dans ce cas, il faut ajouter un formulaire et un lien pour valider la requête. Il faut prendre en considération l'ensemble de ces changements induits car ceux-ci peuvent avoir un impact sur des critères de qualité et donc sur la qualité de l'application.

L'approche « forward » se charge de sélectionner les meilleures transformations, parmi une liste de transformations potentielles, dans le but d'augmenter le niveau de qualité d'une application. Chacune des transformations a un impact théorique sur les critères de qualité et le MQ permet de déterminer quelles sont les meilleures transformations en évaluant les modifications sur les critères. Le problème consiste donc à sélectionner une séquence de transformations qui donne le meilleur niveau de qualité tout en minimisant l'effort d'implémentation.

Pour ce faire, il faut d'abord déterminer l'impact des transformations sur l'ensemble des critères de qualité et, ensuite, le coût associé à chaque transformation pour représenter l'effort d'implémentation. Chacune des séquences possibles de transformations possède donc un coût variable dépendant des transformations qui la compose. Trouver la séquence adéquate peut être considérée comme un processus d'optimisation tenant compte de ce coût et de l'amélioration de qualité.

L'approche « backward » revient donc à déterminer quelles transformations sont capables de modifier les critères indiqués par le MQ pour atteindre un niveau de qualité minimal fixé. Chacune des transformations possède un coût d'implémentation et doit correspondre à une tâche bien définie d'un dévelop-peur.

Dans certains cas, le modèle peut même suggérer des valeurs pour chacun des critères de qualité concernés. Cette suggestion est possible si le MQ permet ce type de raisonnement (c'est-à-dire d'inférence arrière). Comme nous l'avons vu dans l'état de l'art, les modèles de qualité bayésiens permettent ce type d'inférence. En effet, les valeurs des noeuds d'un réseau s'adaptent en fonction des noeuds instanciés. Il est donc possible d'exploiter l'inférence de celui-ci pour prédire les valeurs des critères de qualité pour un niveau de qualité donné.

Cette approche revient donc à déterminer quelles sont les meilleures transformations modifiant les critères de qualité indiqués par le modèle en minimisant l'effort d'implémentation.

Ces deux méthodes ont pour objectif de sélectionner des transformations à apporter sur une application Web dans le but d'améliorer sa qualité. La

première méthode est un problème de recherche parmi des transformations potentielles tandis que la deuxième est un problème recherchant des transformations modifiant des critères déterminés. Dans le cadre de cette thèse, la première approche a été privilégiée pour les raisons suivantes.

- Bien que la seconde approche indique les critères responsables de la mauvaise qualité de l'application, et dans le cas des MQ bayésiens, les valeurs des critères de qualité pour un niveau de qualité donné, rien ne garantit la possibilité de trouver des transformations applicables sur l'application Web modifiant la qualité comme le modèle le suggère. En effet, garantir cette contrainte peut donc s'avérer difficile voir impossible pour un niveau de qualité donné. Car, théoriquement, il faut que les transformations modifient seulement les critères déterminés par le modèle et ceux-là uniquement. Dans le cas contraire, il est nécessaire de prendre en compte les effets de bord engendrés par les transformations (c'est-à-dire modifier d'autres critères de qualité), ce problème pouvant être complexe à résoudre.
- De plus, dans le cas des réseaux bayésiens, une augmentation du niveau de qualité influence la valeur des noeuds représentant des critères de qualité, ceux-ci, par définition, sont des probabilités. Il faut donc convertir ces probabilités en valeurs de métriques. Le modèle pourrait donc suggérer des valeurs de métriques n'ayant aucun sens pour une application Web, comme par exemple supprimer tous les liens d'une page Web. Il est également possible qu'une probabilité ne soit pas convertible en métrique, ce qui contraint à définir l'ensemble des variations possibles. De plus, il faut pourvoir gérer les potentiels conflits entre les différentes valeurs proposées.

La méthode « forward » pose de nombreuses contraintes et problèmes complexes à résoudre tandis que l'approche de sélection de séquences de transformations reste relativement simple. En effet, l'avantage de cette dernière réside dans la définition de transformations potentielles applicables sur une application Web. Il suffit donc de déterminer les impacts des transformations sur les métriques et l'effort associé à l'implémentation des transformations.

4.3 Approche générale de recommandations

Notre approche générale de recommandations consiste en une méthode qui sélectionne, à partir d'un ensemble de transformations potentielles, la meilleure séquence à appliquer sur une page Web donnée dans le but d'améliorer son niveau de qualité pour un moindre effort de développement. L'évaluation de la qualité des modifications est déterminée par un MQ et l'effort est calculé en utilisant la perception d'experts du Web.

Cette approche se veut générale dans le sens où elle est peut pour s'adapter à tout type de MQ, comme nous le verrons par la suite. Dans le cadre de ce mémoire, nous privilégions cependant un MQ probabiliste pour ces nombreux avantages énoncés dans le chapitre 2.

Pour être efficace, cette approche de recommandations doit être réalisable automatiquement, ou du moins le plus automatiquement possible. En effet, l'identification des modifications à apporter sur une application Web doit s'inscrire dans un processus d'amélioration de la qualité et ne doit pas demander plus de temps que l'implémentation des transformations. L'idéal est de créer un outil responsable de suggérer automatiquement ces recommandations dans le but de faciliter la tâche du développeur désirant améliorer la qualité de son application. De plus, cette approche permettrait de rendre compte des problèmes de qualité d'une application Web. En effet, les développeurs n'en sont pas toujours conscients. La création d'un tel outil est abordé dans le chapitre 7 « Étude de cas ».

La figure 4.1 schématise notre approche de recommandations. Celle-ci est divisée en deux processus :

- Un processus d'évaluation responsable de l'évaluation de la qualité d'une application Web donnée. Cette évaluation est réalisée à partir de valeurs de métriques extraites de l'application et utilise un MQ Web qui calcule un score, c'est-à-dire une estimation précise du niveau de qualité de l'application.
- Un processus de recommandations sélectionnant, parmi un ensemble de transformations potentielles candidates ($T_{candidates}$), la meilleure séquence ($ST_{Meilleures}$) à appliquer, c'est à dire une séquence qui donne le meilleur niveau de qualité tout en minimisant l'effort d'implémentation.

Nous allons détailler ces deux processus dans le but de montrer comment ceux-ci permettent l'amélioration de la qualité d'une application Web.

4.3.1 Processus d'évaluation de la qualité

Le processus d'évaluation est responsable d'évaluer la qualité d'une application Web. Ce processus consiste en deux étapes : l'extraction de valeurs de métriques et l'évaluation de la qualité.

Extraction de valeurs de métriques

L'extraction fournit, à partir d'une application Web, les entrants nécessaires pour la phase d'évaluation. En effet, l'évaluation de la qualité n'est possible

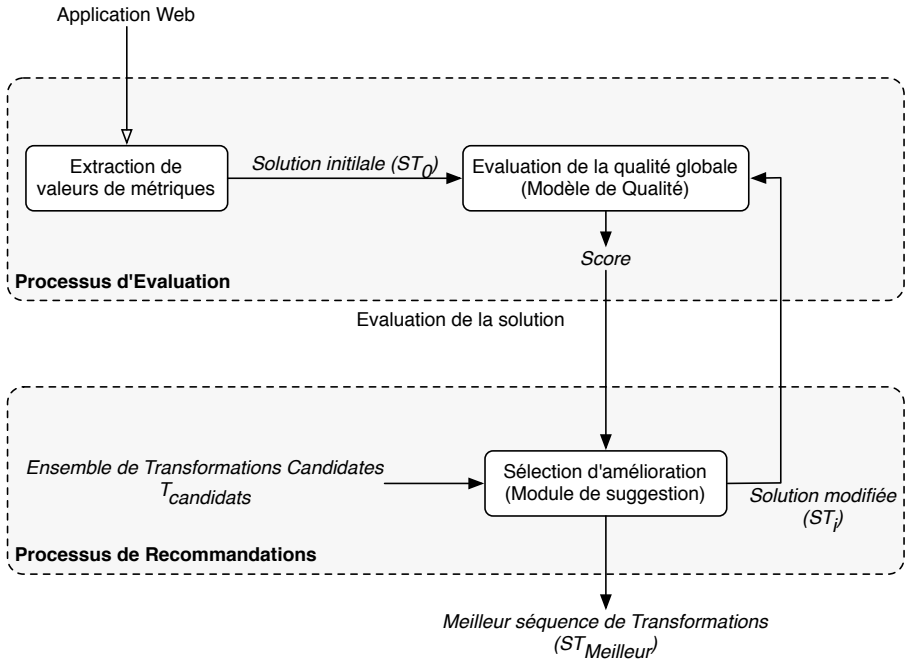


FIGURE 4.1 – Approche de recommandations pour l'amélioration de la qualité des applications Web

qu'a partir de critères mesurables extraits d'une application Web. La principale difficulté de cette étape est l'automatisation. Une extraction automatique est, en effet, nécessaire pour que notre approche de recommandations soit réalisable automatiquement. Nous aborderons cette problématique dans le chapitre 7 « Étude de cas ».

Évaluation de la qualité

L'évaluation est réalisée par un MQ qui doit, d'une part, évaluer la qualité d'une application Web dans son état initial (sans aucune modification) et, d'autre part, évaluer les effets des transformations sur la qualité de l'application. Pour ce faire, le MQ utilise les valeurs de métriques extraites d'une application Web donnée et calcule une estimation du niveau de qualité de l'application.

Notre approche de recommandations considère le MQ comme une boîte noire qui calcule un score à partir d'un ensemble de valeurs de métriques. Ce score représente le niveau de qualité de l'application. Formellement, si m représente un vecteur de valeurs de métriques $\{m_1, m_2, ...\}$ réalisées sur une application Web, le MQ peut être perçu comme une fonction q calculant un

score :

$$score = q(m) = q(m_1, m_2, ..., m_l) \tag{4.1}$$

L'approche boîte noire permet de ne pas s'intéresser à la manière dont le modèle calcule le score mais bien au résultat lui-même. Cette approche générale offre donc l'avantage d'utiliser n'importe quel type de MQ Web si ce dernier utilise des métriques pour calculer un score représentant le niveau de qualité d'une application Web.

Dans le cadre de cette thèse, notre approche de recommandations est illustrée en utilisant un modèle de qualité probabiliste capable d'évaluer une caractéristique de la qualité. Ce modèle évaluant la navigabilité d'une page Web est détaillé dans le chapitre 5 : ≪ Processus d'évaluation de la qualité : application à la navigabilité ≫.

L'utilisation d'un MQ offre également l'avantage d'automatiser la perception d'experts pour juger le plus objectivement possible la qualité d'une application Web. Dans notre approche de recommandations, le MQ permet de sélectionner la meilleure séquence de transformations. Pour ce faire, il doit prendre en compte l'ensemble des modifications engendrées par les transformations y compris en présence d'informations contradictoires. En effet, une transformation qui ajoute un moteur de recherche ne se contente pas d'ajouter des fonctionnalités de rechercher, mais ajoute également un formulaire et un lien pour valider la requête. Ces ajouts pourraient nuire à la qualité de la page. Il faut dès lors tenir compte de l'ensemble des fonctionnalités ajoutées lors de l'évaluation de la qualité. Le MQ permet donc d'évaluer les transformations en prenant en compte l'ensemble des modifications. Il permet donc, dans ce cas, de déterminer si l'ajout d'un moteur de recherche améliore ou pas la qualité de l'application.

4.3.2 Processus de recommandations

Le processus de recommandations est responsable de la sélection de la meilleure solution, c'est-à-dire la séquence de transformations optimales à appliquer sur une application Web dans le but d'améliorer sa qualité. Cette sélection est réalisée à partir d'un ensemble de transformations potentielles candidates qui influencent le niveau de qualité. Ce processus utilise un MQ qui évalue les effets des transformations sélectionnées sur la qualité de l'application. Ce modèle permet donc de comparer l'amélioration de qualité apportée d'une solution par rapport à une autre. Le MQ utilise des métriques pour évaluer la qualité d'une application Web ($q(m)$ voir équation 4.1), par conséquent, les transformations doivent être définies en fonction de leurs effets sur ces métriques.

Une solution optimale est une solution qui prend également en compte l'effort d'implémentation des transformations par rapport au gain de qualité

apporté. En d'autres mots, le processus de recommandations se charge de sélectionner la séquence de transformations qui maximise le score de qualité $q(m)$ tout en minimisant cet effort.

Après avoir présenté la notion de transformations candidates, nous montrons comment le modèle de qualité évalue la qualité d'une solution. Ensuite, nous précisons comment sélectionner les transformations optimales. Pour terminer, nous abordons la recherche d'une solution optimale.

Transformations candidates

Soit un ensemble de transformations potentielles candidates à l'amélioration de qualité ($T_{candidates}$). Chaque transformation appartenant à $T_{candidates}$ correspond à une modification simple et à une tâche bien définie pour un développeur.

$$T_{candidates} = \{t_1, t_2, ..., t_o\} \tag{4.2}$$

Une transformation candidate t_i est caractérisée par le couple d'éléments $t_i =< tr\_met_i, coût_i >$ où tr_met_i représente les effets de la transformation i sur des valeurs de métriques m du MQ et $coût_i$ représente l'effort engendré par l'implémentation de la transformation i sur une application.

tr_met_i peut donc être considérée comme une fonction qui détermine les effets de la transformation t_i sur les valeurs de métriques $\{m_1, ..., m_l\}$ du MQ produisant de nouvelles valeurs de métriques $\{m'_1, ..., m'_l\}$. Étant donné qu'une transformation peut avoir des effets sur plus d'une valeur de métriques, cette fonction peut s'écrire de la façon suivante :

$$tr\_met_i(t_i\ , \{m_1, ..., m_l\}) = \{m'_1, ..., m'_l\}$$

Mesure de l'amélioration de la qualité

Une solution du processus de suggestion est une séquence ST de transformations t_i ordonnées ($ST = (t_1, t_2, ..., t_n)$) où t_i appartient à l'ensemble $T_{candidates}$ ($t_i \in T_{candidates}$). L'impact d'une telle séquence de transformations sur une application Web peut s'exprimer sous la forme de la fonction $transform\_seq$ qui modifie le vecteur de valeurs de métriques m en cumulant les effets des transformations $t_1, t_2, ..., t_n$ dans l'ordre.

Cette fonction calcule de nouvelles valeurs métriques m' à partir des transformations contenues dans la séquence ST et de valeurs de métriques m. Elle se charge donc d'appliquer successivement chacune des fonctions tr_met_i qu'elle contient sur les métriques :

$$\begin{aligned} m' &= transform\_seq(ST, m) \\ &= tr\_met_n(t_n, \cdots tr\_met_2(t_2, tr\_met_1(t_1, m)) \end{aligned}$$

Pour évaluer la qualité d'une page modifiée par la séquence ST, il suffit de passer les valeurs m' au MQ. Comme le montre l'équation 4.1 ci-dessus, le modèle évalue la qualité $q(m')$ avec les valeurs de métriques m'.

Sélection des solutions

Le processus de recommandations doit être capable d'identifier la meilleure solution qui maximise l'amélioration de la qualité tout en minimisant l'effort d'implémentation. De façon formelle, l'objectif est donc de trouver la séquence $ST_{meilleure}$:

$$\forall(ST_i) : f(transform\_seq(ST_{meilleure}, m)) > f(transform\_seq(ST_i, m)) \tag{4.3}$$

Sélectionner cette meilleure solution est réalisée par la fonction f qui permet d'évaluer la « valeur » d'une solution par rapport à une autre en prenant en compte l'effort d'implémentation et la qualité d'une solution. Cette fonction doit donc combiner deux paramètres possédant des unités différentes sans en favoriser un plus que l'autre. Cette problématique est abordée dans le chapitre 6 « Processus de recommandations : application à la navigabilité ».

Recherche d'une solution optimale

Trouver la solution optimale est un problème de recherche. Naïvement, il est nécessaire de considérer l'ensemble des solutions possibles pour trouver la séquence optimale. Tester toutes les séquences possibles est réalisable dans un temps raisonnable si et seulement si l'espace de recherche n'est pas trop grand. Ce dernier dépend directement du nombre de transformations à considérer et peut devenir potentiellement infini si une transformation peut être appliquée plusieurs fois.

De plus, étant donné qu'il n'y a pas de relation directe entre une transformation et ses effets sur le MQ, il est difficile de limiter cet espace. En effet, le MQ est perçu comme une boîte noire qui évalue la qualité à partir de métriques. Bien qu'une transformation $T_{candidates}$ modifie la qualité d'une application Web, celle-ci peut aussi bien améliorer que diminuer la qualité.

Dans le chapitre 6, nous présentons l'ensemble des transformations candidates utilisées pour améliorer le niveau de navigabilité d'une page Web, nous montrons que cette recherche d'une solution ne peut être réalisée dans un temps polynomial. Ceci nous a contrains à utiliser une algorithme heuristique guidant l'exploration dans un large espace de recherche pour trouver une solution dans un temps raisonnable, comme nous l'avons vu dans l'état de l'art.

Chapitre 5

Processus d'évaluation de la qualité : application à la navigabilité

5.1 Introduction

L'objectif de ce mémoire est de présenter une approche générale permettant d'améliorer la qualité d'une application Web en utilisant un MQ. Ce modèle est utilisé par le « Processus d'évaluation » pour fournir un score estimant le niveau de qualité de l'application et par le « Processus d'amélioration » pour évaluer les effets des séquences de transformations proposées sur la qualité comme nous l'avons présenté dans le chapitre 4.

Dans le cadre de ce mémoire, nous avons choisi d'illustrer notre approche à la navigabilité, une caractéristique de la qualité. Dans ce chapitre, nous présentons le MQ probabiliste défini par Malak [19] et plus particulièrement le modèle qui évalue la navigabilité d'une page Web.

5.2 Un modèle de qualité probabiliste

Un MQ Web doit donner une bonne indication du niveau de qualité d'une application Web à partir de métriques extraites de celle-ci. Cette indication doit représenter la perception d'un expert de la qualité du Web. Notre approche de recommandations impose la comparaison de qualité de différentes suggestions, ce qui est possible si le MQ fournit un score représentant le niveau de qualité de l'application. Ce score peut donc être utilisé lors du processus d'amélioration pour sélectionner la meilleure séquence de transformations.

Malak [19] propose une méthodologie qui permet d'évaluer la qualité des

applications Web. Son approche explique la construction d'un MQ probabiliste en utilisant les réseaux bayésiens. Son modèle fournit un score qui correspond à la probabilité qu'un utilisateur considère la qualité de l'application satisfaisante. Ce score est calculé à partir de différents critères de qualité mesurés (métriques) extraits d'une application Web et utilisés comme entrant du modèle.

Son approche a été illustrée et validée en définissant un réseau qui évalue la navigabilité d'une page Web. Notre approche de recommandations utilise ce modèle de navigabilité pour illustrer notre démarche. Ce MQ calcule donc un score qui représente la probabilité qu'un utilisateur considère la navigabilité d'une page satisfaisante. Formellement, ce modèle est donc vu comme une fonction q_{NAV} qui évalue la navigabilité d'une page Web à partir du vecteur de valeurs de métriques m. Cette fonction correspond à la fonction 4.1 appliquée au cadre de la navigabilité.

$$score = q_{NAV}(m) = q_{NAV}(m_1, m_2, ..., m_l) \tag{5.1}$$

Nous allons expliciter les concepts nécessaires à la compréhension du modèle de navigabilité utilisé dans le cadre de ce mémoire. Nous n'allons cependant pas détailler la méthodologie qui permet de construire un réseau bayésien évaluant la qualité d'une application Web. Le lecteur est invité à consulter la thèse de Malak pour plus de détails à ce sujet.

5.3 Un modèle de qualité bayésien évaluant la navigabilité d'une page Web

5.3.1 Le réseau bayésien

Le MQ bayésien fournit une évaluation de la qualité (bonne ou mauvaise) en fonction des valeurs des noeuds d'entrée du modèle. Ces dernières sont issues de la conversion en probabilité de valeurs métriques extraites d'une page Web. Par conséquent, les valeurs de ces noeuds d'entrée influencent de façon positive ou négative les valeurs des noeuds fils (considérés comme noeuds intermédiaires), ceux-ci influençant également leurs fils, et ainsi de suite jusqu'à fournir une estimation du niveau de qualité.

La figure 5.1 illustre le réseau bayésien responsable de l'évaluation de la navigabilité d'une page Web.

Le niveau de navigabilité d'une page est représenté par le noeud navigabilité qui contient 2 états. L'un représente la probabilité qu'un utilisateur considère la navigabilité d'une page satisfaisante (état « Good ») et, l'autre, la probabilité qu'elle ne le soit pas (état « Bad »).

Dans le cadre de ce mémoire, seule la valeur qui représente une navigabilité satisfaisante est utilisée et correspond au *score* de navigabilité repris dans l'équation 5.1.

Comme le montre la figure 5.1, ce noeud de navigabilité est influencé par ses noeuds fils, c'est-à-dire par les noeuds représentant la localisation (noeud « Locate »), le temps de téléchargement (noeud « DownloadTime ») et la liaison entre pages (noeud « Revisit »). La localisation est elle-même influencée par les feebacks utilisateurs (noeud « UserFeedback ») et les éléments de navigation (noeud « NavigationMechanisms »). Ce raisonnement est valable jusqu'aux noeuds sans parents représentant les critères extraits à partir d'une page Web. La signification de chacun de ces noeuds est présentée dans la thèse de Malak.

5.3.2 Les tables de probabilité

Comme présenté dans le chapitre 3, chaque noeud possède des paramètres qui sont contenus dans une table de probabilités. L'assignement des valeurs aux tables dépend de la variable représentée par le noeud.

Noeuds intermédiaires

Les valeurs des tables des noeuds intermédiaires ont été déterminées à partir d'études dans le domaine et d'avis d'experts et peuvent être réajustées en fonction de l'application à évaluer [19]. Ces tables des noeuds intermédiaires modélisent la relation d'incertitude avec ses noeuds parents par l'intermédiaire d'une table de probabilités conditionnelles.

La figure 5.2 reprend la table de probabilités du noeud « NavigationOptions » représentant les options de navigabilité. Ce noeud est influencé par le noeud « LinkToHome » représentant le lien vers la page d'accueil et celui « BackButton » représentant le retour par un bouton. Le détail des autres tables des probabilités se trouvent en annexe.

Noeuds d'entrée

Les noeuds d'entrée du réseau représentent des critères mesurés (également appelés métriques) à partir d'une application Web donnée. Les tables de probabilités de ces noeuds d'entrée sont obtenues en convertissant les valeurs mesurées des critères en probabilité. Le réseau évaluant la navigabilité d'une page contient 13 entrées, il y a donc 13 tables de probabilités représentant 13 critères.

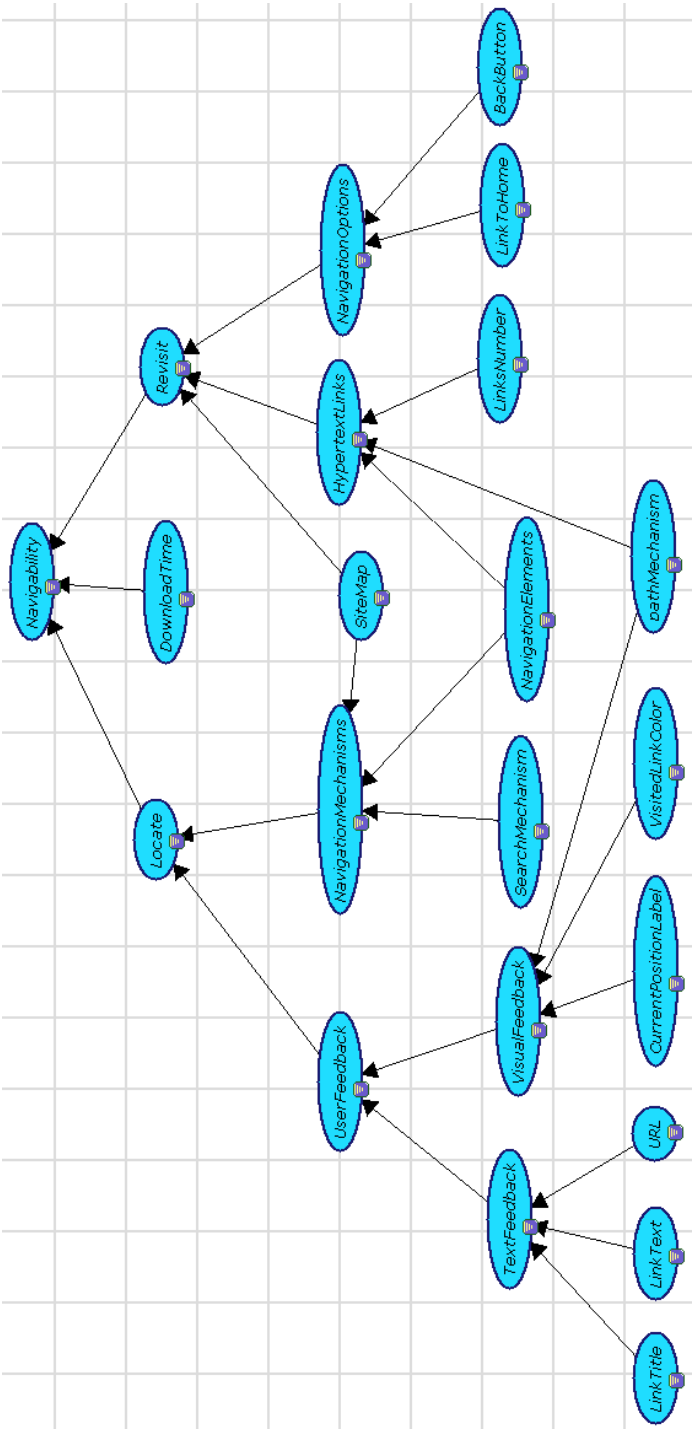
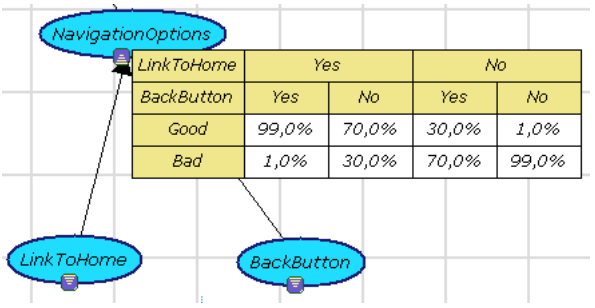


FIGURE 5.1 – Réseau bayésien évaluant de navigabilité d'une page Web



LinkToHome	Yes		No	
BackButton	Yes	No	Yes	No
Good	99,0%	70,0%	30,0%	1,0%
Bad	1,0%	30,0%	70,0%	99,0%

FIGURE 5.2 – Extrait d'une table de probabilités du réseau bayésien évaluant la navigabilité

L'ensemble de ces critères mesurables identifiés sont de trois types différents : binaires (B), entiers (E) et décimaux (D).

- Les critères binaires représentent l'absence ou la présence d'un élément sur une page Web. Par exemple, le critère « Présence d'un menu » indique la présence ou l'absence d'un menu.
- Les critères entiers sont issus soit d'un comptage ou d'une mesure. Par exemple, la valeur du critère « Nombre de liens par page » est obtenue en comptant les liens présents sur la page Web et la valeur du critère « Temps de téléchargement de la page » est obtenue en mesurant le temps nécessaire pour télécharger la page Web.
- Les critères décimaux représentent le pourcentage d'un élément de la page. Par exemple, le critère « Présence d'un titre de lien » représente le pourcentage de titres de liens pour tous les liens d'une page Web.

Une description des 13 critères ainsi que les noms des noeuds associés et leurs types (T) se retrouve détaillée dans le tableau 5.1.

Le processus de conversion responsable de calculer les valeurs de métriques en probabilité dépend du type du noeud :

Les tables de probabilités des noeuds représentant des critères binaires contiennent deux états, un pour l'absence du critère avec une probabilité de 1%, l'autre pour la présence avec une probabilité de 99%. Par exemple pour une page possédant un moteur de recherche, la table de probabilités à priori est illustrée par le tableau 5.2.

Les tables de probabilités pour les critères décimaux sont remplies en fonction du pourcentage d'éléments dans une page. La probabilité de chaque état

Tableau 5.1 – Description des critères d'entrée et noeuds associés du réseau bayésien pour l'évaluation de la navigabilité

Critère d'entrée	Nom du Noeud	T
Présence d'un mécanisme de recherche (Rech) Description : Dans la page, existe-t-il oui ou non un mécanisme de recherche ?	SearchMechanism	B
Présence d'un plan de site (Plan) Description : Dans la page, existe-t-il oui ou non un lien vers un plan ou une carte de site ?	SiteMap	B
Présence d'un URL relatif (URL) Description : L'URL de la page est-elle oui ou non relatif à la page ?	URL	B
Présence d'un lien vers la page d'accueil (Accueil) Description : Dans la page, y-a-t-il oui ou non un lien vers la page d'accueil ?	LinkToHome	B
Présence d'un menu (Menu) Description : Dans la page, existe-t-il oui ou non une barre ou un menu de navigation ?	NavigationElements	B
Présence d'un indicateur de la position courante (PosCour) Description : Dans la page, la position courante est-elle oui ou non indiquée (c'est-à-dire le lien vers la page courante grisaillé, accentué ou inactif) ?	CurrentPositionLabel	B
Présence d'un indicateur de chemin (Chemin) Description : Dans la page, existe-t-il oui ou non un indicateur de chemin au niveau de la barre de navigation ?	pathMechanism	B
Présence d'un changement de couleur du lien visité (CoulLiens) Description : Dans la page, les liens visités changent-ils oui ou non de couleurs ?	VisitedLinkColor	B
Présence d'un bouton Retour Actif (Retour) Description : Dans la page, les boutons précédent et suivant restent-ils actifs oui ou non ?	BackButton	B
Nombre de liens par page (Liens) Description : Mesure du nombre total de liens de la page.	LinksNumber	E
Temps de téléchargement de la page (Tps) Description : Temps (ms.) nécessaire pour télécharger l'entièreté de la page.	DownloadTime	E
Pourcentage de textes de liens significatifs (Textes-Liens) Description : Pourcentage ($[0, 1]$) de liens représentatif. Liens représentatifs = liens sous forme de texte significatif - Liens sous forme des mots comme « learn more », « more », « previous », etc.	LinkText	D
Pourcentage de titres de liens (TitresLiens) Description : Pourcentage ($[0, 1]$) de liens avec de titres.	LinkTitle	D

Tableau 5.2 – Exemple de table de probabilités à priori pour un noeud d'entrée de type binaire.

≪ **Oui** ≫	0.99
≪ **Non** ≫	0.01

est donc déterminée à partir des pourcentages calculés. Par exemple, dans un page Web donnée, si le nombre de liens possédant des titres est de 67%, une probabilité de 0.67 est attribuée à l'état ≪ Avec titre ≫ et 0.33 pour l'état ≪ Sans titre ≫, comme le montre le tableau 5.3.

Tableau 5.3 – Exemple de table de probabilités à priori pour un noeud d'entrée de type décimal.

≪ **Avec titre** ≫	0.67
≪ **Sans titre** ≫	0.33

Pour les tables de probabilités représentant des critères de type entier, il est nécessaire de convertir les valeurs de métriques en utilisant la logique floue. En effet, la valeur de ces types de métriques est différente d'une application à l'autre et le nombre de valeurs possibles peut potentiellement être infini, comme par exemple, pour le nombre de liens dans une page.

Comme expliqué dans le chapitre 3, pour faciliter la définition en probabilités, ces valeurs sont transformées en variables discrètes ayant un nombre limité de valeurs. Ce processus de conversion peut être réalisé par l'intermédiaire de la logique floue qui remplace les différentes valeurs d'un critère par un ensemble de fonctions qui représentent le degré d'appartenance de chaque valeur aux différents classes floues (souvent ≪ Grand ≫, ≪ Moyen ≫ et ≪ Petit ≫) [19]. Les probabilités des tables pour un critère mesuré sur une page Web sont obtenues en calculant le pourcentage d'appartenance de la métrique à la classe.

Par exemple, la table de probabilités pour le critère représentant le nombre de liens d'une page peut contenir les valeurs présentes dans le tableau 5.4.

Tableau 5.4 – Exemple de table de probabilités pour un noeud d'entrée de type entier.

≪ **Petit** ≫	0.01
≪ **Moyen** ≫	0.66
≪ **Grand** ≫	0.33

Chapitre 6

Processus de recommandations : application à la navigabilité

6.1 Introduction

Dans ce chapitre, nous détaillons notre contribution qui permet de sélectionner la meilleure séquence de transformations $ST_{Meilleure}$ comme présenté dans le chapitre 4 avec le processus de recommandations. Pour illustrer ce processus, nous présentons un exemple de recommandations appliqué à la navigabilité d'une page Web en utilisant le modèle de navigabilité défini dans le chapitre 5 « Processus d'évaluation de la qualité : application à la navigabilité ».

Ce chapitre est divisé en deux parties. Dans un premier temps, nous présentons l'ensemble des transformations candidates et leurs effets définis en fonction des métriques utilisées par le modèle de navigabilité. Ensuite, nous illustrons la sélection de la meilleure séquence de transformations qui améliore le niveau de navigabilité de la page Web. Cette sélection est réalisée par l'intermédiaire d'un algorithme heuristique : le recuit simulé. Nous montrons comment cet algorithme est adapté à notre problème de sélection.

6.2 Les transformations pour la navigabilité

Pour améliorer la qualité d'une application Web, il est nécessaire d'identifier l'ensemble des transformations potentielles candidates ($T_{candidates}$) ainsi que leurs effets sur les métriques utilisées par le modèle de qualité.

Dans le cadre de ce mémoire, nous illustrons notre approche générale de recommandations pour améliorer la navigabilité d'une page Web. Nous de-

vons donc définir un ensemble de transformations qui influence le niveau de navigabilité d'une page et leurs effets sur les métriques utilisées par le modèle de navigabilité présenté dans le chapitre précédent. Soit $T_{NAV} \subset T_{candidates}$ est l'ensemble de transformations candidates disponibles qui influencent la navigabilité d'une page Web. Cet ensemble contient des changements généraux de réfactoring ou des changements spécifiques liés à la navigabilité du Web (comme par exemple l'ajout d'un moteur de recherche).

Pour rappel, toute transformation appartenant à l'ensemble T_{NAV} doit correspondre à un changement bien défini pour un développeur et doit rester cohérente par rapport à ce qu'elle modifie. Ceci signifie, par exemple, que la transformation qui ajoute un moteur de recherche ne se contente pas uniquement de modifier la valeur de métrique concernant la présence d'un moteur de recherche mais modifie également toutes les autres métriques concernées par l'ajout. Dans ce cas, l'ajout d'un tel moteur modifie la page Web en ajoutant un formulaire et un lien pour valider la requête. Cette modification influence donc la valeur d'autres métriques comme le nombre de liens qui augmente de un. Il est important de prendre en considération l'ensemble de ces changements car ceux-ci peuvent avoir un impact sur les valeurs de métriques et donc une influence sur la qualité de la page.

L'ensemble T_{NAV} représente une « base de connaissance » pour améliorer la navigabilité d'une page. Cet ensemble est extensible et adaptable. Il est donc possible d'ajouter ou de supprimer des transformations de cet ensemble.

Dans le cadre de ce mémoire, nous avons identifié 12 types de transformations définies en fonction de leurs effets sur les valeurs de métriques utilisées par le modèle de navigabilité. Ces critères mesurés sont présentés dans le tableau 5.1 du chapitre précédent.

Le tableau 6.1 reprend ces transformations, leurs types (T) et leurs effets définis par rapport aux métriques.

Trois types de transformations ont été identifiés : unaire (U), binaire (B) et multiple (M).

- Les transformations unaires affectent une seule métrique de type décimal. Elles sont responsables de la correction de problèmes comme corriger les titres ou les textes de liens.
- Les transformations binaires ont pour objectif de changer la valeur des métriques de type binaire. Ces dernières ont généralement des effets secondaires affectant d'autres métriques. Dans la plupart des cas, ces effets concernent le nombre de liens sur une page Web.
- Les transformations multiples affectent une seule métrique et peuvent être répétées plusieurs fois. Comme par exemple la transformation qui

modifie la taille de la page en ajoutant ou supprimant des liens.

Tableau 6.1 – Ensemble des transformations (T_{NAV}) et leurs effets sur les critères (métriques) présentés dans le tableau.5.1

#	Transformations	T	Effets sur les métriques
1	Ajout/Suppression d'un moteur de recherche	B	Liens +/- 1, Rech = oui/non
2	Ajout/Suppression d'un plan du site	B	Liens +/- 1, Plan = oui/non
3	Ajout/Suppression d'un URL relatif	B	URL = oui/non
4	Ajout/Suppression d'un lien vers la page d'accueil	B	Liens +/- 1, Accueil = oui/non
5	Ajout/Suppression d'un menu	B	Liens +/- b^*, Menu = oui/non
6	Ajout/Suppression d'un indicateur de position courante	B	PosCour = oui/non
7	Ajout/Suppression d'un indicateur de chemin	B	Liens +/- c^*, Chemin = oui/non
8	Ajout/Suppression du changement de couleur des liens visités	B	CoulLiens = oui/non
9	Activation/Désactivation du bouton de retour arrière	B	Retour = oui/non
10	Diviser/fusionner une page	M	Liens $\div$/x 2 +/- d^*
11	Correction des textes de liens	U	TextesLiens = 100%
12	Correction/Suppression des titres de liens	U	TitresLiens = 100% / 0 %

Aucune transformation cohérente n'a été identifiée pour influencer la métrique « Temps de téléchargement de la page ». Celle-ci dépend fortement de paramètres non contrôlables directement, comme la vitesse de la connexion. Il n'est donc pas possible qu'une modification simple, correspondant à une tâche bien définie d'un développeur, modifie cette métrique.

Certaines transformations comme « Ajout/Suppression d'un indicateur de chemin »,« Ajout/Suppression d'un menu » et « Diviser/fusionner une page » ont un effet sur un nombre variable de liens (dénoté par les variables b^*, c^*, d^*). Le nombre exact de liens dépend des applications Web considérées et doit être évalué empiriquement. Cette problématique sera abordée dans le chapitre 7 lors des expérimentations.

La transformation 10 responsable de diviser ou fusionner une page est soumise à une pré-condition. Celle-ci limite le nombre de divisions pour éviter que le nombre de liens ne soit trop faible. En effet, il devient absurde de di-

viser une page Web si le nombre de liens tend vers 0. Le nombre minimum est une variable devant être évaluée empiriquement. Cette problématique sera également abordée dans le chapitre 7.

Chaque transformation est associée à un « coût » modélisant le niveau d'effort nécessaire à l'implémentation. En effet, notre approche de recommandations tient compte de l'effort. Pour ce faire, un nombre entre un et dix est associé à chacune des transformations. Plus ce nombre est important, plus l'effort est élevé. Ce coût correspond à l'estimation de l'effort d'implémentation relatif d'une transformation par rapport à une autre et est fixé en utilisant l'avis d'experts de la qualité du Web. Ce nombre est utilisé pour sélectionner la meilleure séquence comme nous l'illustrons dans la section « Sélection des meilleures transformations » ci-dessous. Le tableau 6.2 détaille ces coûts.

Tableau 6.2 – Ensemble des niveaux d'effort relatifs (coûts) pour les transformations T_{NAV}

#	Transformations	Coûts
1	Ajout/Suppression d'un moteur de recherche	6
2	Ajout/Suppression d'un plan du site	3
3	Ajout/Suppression d'un URL relatif	2
4	Ajout/Suppression d'un lien vers la page d'accueil	1
5	Ajout/Suppression d'un menu	6
6	Ajout/Suppression d'un indicateur de position courante	5
7	Ajout/Suppression d'un indicateur de chemin	6
8	Ajout/Suppression du changement de couleur des liens visités	2
9	Activation/Désactivation du bouton de retour arrière	1
10	Diviser/fusionner une page	8
11	Correction des textes de liens	5
12	Correction/Suppression des titres de liens	6/4

6.3 Sélection des meilleures transformations

6.3.1 Un problème complexe

Une approche naïve pour suggérer des séquences de transformations serait de considérer individuellement l'ensemble des arrangements possibles. Il suffirait dès lors de sélectionner la séquence qui améliore le mieux la navigabilité pour un coût minimal. Bien que la petite taille de l'ensemble T_{NAV} permette une exploration de l'ensemble des possibilités en un temps raisonnable, cette approche n'est pas évolutive dans le cas d'un ensemble plus grand. En effet, cette recherche consiste à trouver une séquence à partir de l'ensemble de transformations. Sans considérer l'ordre, ceci peut revenir à parcourir l'ensemble des parties de l'ensemble des transformations. Or, cette recherche d'une solution

de façon exhaustive est un problème combinatoire. La cardinalité de ce dernier est de 2^m où m correspond à la taille de l'ensemble. Si on considère l'ordre pour chacun des sous-ensembles (en effet, on recherche une séquence), cela augmente la complexité du problème. L'espace de recherche est de $n!$ pour chaque sous-ensemble de taille n.

De plus, il est possible qu'une même transformation soit répétée plusieurs fois dans le cas des transformations multiples. Ceci augmente encore l'espace de recherche et le rend potentiellement infini.

Pour évaluer une solution, il est nécessaire d'utiliser le MQ bayésien. Le temps nécessaire pour trouver la solution optimale doit donc prendre en compte le temps d'évaluation. Or, résoudre un réseau bayésien est un problème NP-Hard [9]. La complexité en temps est exponentielle à la densité du réseau et au nombre d'états des variables aléatoires de chaque noeud. Bien que le faible nombre de noeuds du réseau permet une évaluation exacte (*inférence exacte*) de la navigabilité d'une page dans un temps raisonnable, l'évolutivité du modèle risque de jouer en défaveur du temps d'exécution. Une *inférence approximative* serait donc une alternative à considérer si le temps d'exécution devient trop grand.

6.3.2 L'approche de recommandations par un algorithme heuristique

La complexité du problème, nous permet d'utiliser un algorithme heuristique pour guider la recherche d'une solution d'un grand espace de recherche pour converger vers une solution optimale dans un temps raisonnable. Dans le cadre de ce mémoire, nous utilisons l'algorithme heuristique du recuit simulé adapté à notre problème de recherche d'une séquence de transformations optimales.

L'utilisation de cet algorithme se justifie aussi car l'évaluation d'une solution par le MQ bayésien a une complexité en temps élevée. Un algorithme heuristique, ne parcourant pas l'ensemble des solutions, est donc une bonne alternative.

De plus, notre approche de recommandations se veut générale et applicable à tout type de MQ. Il est donc possible de définir un ensemble de transformations $T_{candidates}$ plus grand (supérieur à $|T_{NAV}|$) qui augmenterait la taille de l'espace de recherche. Nous approche globale justifie également l'utilisation d'un algorithme heuristique. Dans le chapitre 7 « Étude de cas », nous aborderons dans quelle mesure notre méthode est adaptable à des problèmes de plus grande complexité.

La chapitre 3 ayant présenté l'algorithme du recuit simulé, cette partie illustre la manière dont nous l'avons adaptée pour sélectionner la meilleure

séquence de transformations.

L'espace de solutions

T_{NAV} reprend l'ensemble des transformations candidates qui influencent le niveau de navigabilité d'une page Web quelconque. Les 23 transformations contenues dans cet ensemble ne sont cependant pas applicables sur une page particulière. Une exclusion des transformations non valables permet de diminuer la taille de l'espace de recherche et donc d'augmenter les performances lors de la recherche de la séquence optimale.

Il est, par exemple, inutile de considérer la transformation ajoutant un moteur de recherche si la page en possède déjà un et absurde de corriger les titres de liens si ceux-ci sont corrects.

La recherche d'une solution optimale se limite donc aux transformations valides T_{Page} pour une page spécifique $Page$. Cet ensemble est un sous-ensemble de T_{NAV} ($T_{Page} \subset T_{NAV}$) et varie d'une page à l'autre.

La représentation d'une solution

Une solution, c'est-à-dire une séquence de transformations, peut donc être représentée comme un vecteur s de transformations t_i ($s = (t_1, t_2, ..., t_n)$ avec $t_i \in T_{Page}$) à appliquer sur une page Web $Page$ dans le but d'améliorer sa qualité.

La solution à l'état initial s_0 est une page Web sur laquelle aucune modification n'a été appliquée. Par conséquent, s_0 ne contient aucune transformation ($s_0 = ()$).

La fonction de transition

L'algorithme du recuit simulé utilise une fonction de transition (V) responsable de la génération d'une solution voisine temporaire s' à partir de la solution courante s. Cette génération est réalisée à chaque itération jusqu'à ce qu'une solution finale satisfaisante soit trouvée.

Cette fonction choisit aléatoirement d'ajouter une transformation de T_{Page} à la solution ou d'en supprimer une déjà présente dans le vecteur s. Par conséquent, l'espace de recherche contient l'ensemble des transformations T_{Page} et les transformations de suppression.

L'ajout est cependant soumis à certaines contraintes permettant de préserver la cohérence globale de la solution. À chaque itération, la fonction de transition choisit une transformation t_i appartenant à $T_{voisin} \subset T_{Page}$ où $t_i \in T_{voisin} \wedge \mathit{cohérent}(s, t_i)$. La fonction $\mathit{cohérent}(s, t_i)$ s'assure, d'une part, de retourner une solution cohérente et permet également de réduire l'espace

de recherche.

Deux types de situations peuvent conduire à des séquences incohérentes. L'ajout d'une transformation redondante ou l'ajout d'une transformation opposée.

Transformation redondante : Il est redondant d'appliquer deux fois des transformations affectant des valeurs discrètes de métriques. Deux instances d'une transformation unaire ou binaire dans le vecteur de solution ne sont donc pas permises. Par exemple, il ne serait pas cohérent d'avoir 2 transformations qui ajoutent un moteur de recherche dans le vecteur de solution, une suffit.

Transformation opposée : D'autre type de transformations ont des effets opposés, comme par exemple, fusionner et diviser une page. Il est absurde d'avoir une séquence contenant deux types de transformations ayant des effets qui s'opposent. Pour ajouter ce type de transformation, il faut d'abord que son opposé soit supprimé.

Une solution peut, par contre, contenir plusieurs instances de transformations multiples. Il est donc possible de diviser une page plusieurs fois si la pré-condition est respectée (c'est-à-dire si le nombre de liens est suffisamment grand). Cette condition limite le nombre d'instances de transformations multiples dans une séquence.

Une fonction de transition peut être modélisée de différentes manières, mais la littérature [5] conseille d'en choisir une qui ne modifie que légèrement la solution. Notre fonction V ajoute ou supprime une et une seule transformation à chaque itération de l'algorithme.

La fonction objective

La fonction objective *Obj* de l'algorithme du recuit simulé permet d'évaluer la valeur d'une solution. Or, notre approche de recommandations doit prendre en considération 2 paramètres, l'effort nécessaire pour implémenter la solution et son score de qualité évalué par le modèle. Il faut donc combiner ces deux notions pour évaluer la valeur d'une solution par rapport à une autre. La difficulté consiste à combiner de façon équivalente deux paramètres possédant des unités différentes sans pénaliser une solution plus qu'une autre.

Ce problème revient à trouver une fonction *Obj* (comme présenté dans le chapitre 4 avec la fonction 4.3) telle que :

$$\forall(ST_i) : Obj(transform\_seq(ST_{meilleure}, m)) > Obj(transform\_seq(ST_i, m))$$

où ST_i correspond à s_i et $ST_{meilleure}$ à $s_{Meilleure}$ dans l'algorithme du recuit présenté par l'algorithme 2.

Dans ce cas, l'algorithme du recuit simulé doit sélectionner une solution maximisant la valeur de la fonction objective. Une adaptation de l'algorithme 1 (présenté dans le chapitre 3) à un problème de maximisation est nécessaire .

Dans le cadre de ce mémoire, nous avons implémenté différents types de fonctions objectives synthétisées en annexe. Celles-ci utilisent : un seuil de navigabilité, un seuil de coût, une normalisation du coût ; la navigabilité d'une solution, la différence de navigabilité d'une solution par rapport à une autre et d'une solution par rapport à la solution initiale. Nous avons également utilisé des fonctions constantes ou logarithmiques. Cependant, une seule fonction a été retenue car la combinaison des deux paramètres avec des unités différentes n'a pas directement été possible.

En effet, nous avons donc privilégié une fonction objective sélectionnant la meilleure solution $s_{meilleure}$ soumise à une contrainte de coût W telle que :

$$Coût(s_{meilleure}) = Coût(ST_{meilleure}) < W \tag{6.1}$$

La fonction $coût(ST)$ représente le coût global de la séquence de transformations ST. C'est-à-dire que cette fonction additionne les coûts des différentes transformations qui composent la séquence :

$$Coût(ST) = \sum_{1}^{i_l} coût(t_i)$$

où *coût* représente l'effort associé à l'implémentation de la transformation.

Notre fonction objective Obj représentée en 6.2 pénalise une solution si celle-ci ne respecte pas la contrainte de coût W. Une solution avec un meilleur score de navigabilité (par rapport à la solution initiale) qui ne respecte pas la contrainte de coût aura dans tous les cas une moins bonne valeur qu'une solution qui la respecte. Cette fonction objective a l'avantage de ne favoriser aucun des paramètres (l'effort d'implémentation et l'amélioration de navigabilité). Deux solutions sont donc à tout moment comparables.

$$Obj(s) = \begin{cases} 0.5 + Amé\_Nav * 0.5 & \text{si} \quad coût(s) \leq W \\ Amé\_Nav * 0.5 & \text{si} \quad coût(s) > W \end{cases} \tag{6.2}$$

où $Amé\_Nav$ correspond à l'amélioration de la navigabilité, c'est-à-dire à la différence entre le score de navigabilité pour la solution (s) et le score de solution initiale (s_0).

Cette fonction $Obj(s)$ renvoie donc une valeur comprise entre 0 et 1 en favorisant l'amélioration de navigabilité. Par conséquent, nous avons décidé que la fonction $Amé\_Nav$ renvoie une valeur supérieure à 0,5 pour une amélioration

de navigabilité et une valeur inférieure à 0,5 pour une diminution de navigabilité par rapport à la solution initiale s_0. Cette fonction peut s'écrire :

$$Amé\_Nav = \frac{1 + \Delta Nav(s_0, s_i)}{2} \in [0, 1]$$

où $\Delta Nav(s_0, s_i) = |q_{NAV}(s_i) - q_{NAV}(s_0)|$. La fonction q_{NAV} représente le score de navigabilité comme présenté dans le chapitre précédent avec la fonction 5.1.

Les paramètres de l'algorithme

L'algorithme du recuit simulé nécessite l'initialisation de nombreux paramètres comme la température initiale, la température finale, la diminution de la température, la longueur de la température et le critère d'arrêt.

Étant donné que ces paramètres doivent être adaptés en fonction du problème, nous détaillons leurs valeurs dans le chapitre 7 « Étude de cas ».

Simulation de solutions

L'approche que nous avons choisie permet de simuler la création d'une page Web sur laquelle les transformations ont été appliquées. En effet, il n'est pas nécessaire d'implémenter les solutions pour les évaluer. Le MQ permet d'évaluer la navigabilité d'une page à partir des métriques reprises dans le tableau 5.1 via l'équation 5.1. Étant donné que les effets des transformations sont définis en fonction de ces métriques, il suffit d'appliquer les transformations sur les métriques pour évaluer la navigabilité d'une page Web comme si ces transformations avaient été implémentées.

```
RECUIT SIMULÉ (Trs_0 ,Nrs) ;
Choisis s_0                    /* Choix d'une solution initiale */ ;
Trs := Trs_0                 /* Initialisation de la température */ ;
STOP := false ;
s := s_0 ;
s_Meilleure := s_0
                  /* Initialisation de la Meilleure Solution */ ;
while !STOP do
    for i := 1 to Nrs do
        Génère s' ∈ V(s)                /* Génération d'un voisin */ ;
        ΔObj := Obj(s') − Obj(s)          /* Fonction objective */ ;
        if ΔObj ≥ 0 then
            s := s'                        /* Accepte la solution */ ;
        else
            Génère un nombre aléatoire r ∈ [0, 1] ;
            if r ≤ e^(ΔObj/Trs) then
                s := s' /* Accepte avec une petite probabilité */ ;
            end
        end
        if Obj(s') ≥ Obj(s_Meilleure) then
            s_Meilleure := s'        /* Meilleure solution trouvée */ ;
        end
    end
    Trs := α.Trs                        /* Diminue la température */ ;
    if StopCriteria then
        STOP := True ;
    end
end
return s_Meilleure ;
```

Algorithme 2: Algorithme du recuit simulé adapté à un problème de maximisation

Chapitre 7

Étude de cas

Ce chapitre a pour but de montrer la faisabilité de notre approche en l'appliquant sur des cas pratiques.

Dans un premier temps, nous abordons la manière dont nous avons implémenté un outil pour notre approche de recommandations. Ensuite, nous interprétons les résultats de notre approche appliquée sur 15 pages Web.

7.1 Réalisation de l'outil

Notre approche vise à aider un développeur voulant traiter les problèmes de qualité de son application Web. Nous avons donc développé un outil qui permet d'évaluer et suggérer des recommandations semi-automatiquement.

7.1.1 Une application Web pour l'évaluation et l'amélioration

De nombreux sites Web souffrent de problèmes de qualité sans que le développeur en soit toujours conscient. Nous avons choisi de faciliter sa tâche s'il désire évaluer et améliorer la qualité d'une application Web en implémentant un outil à partir des technologies Web.

Le choix d'un outil sous forme d'une application web se justifie car ceci a l'avantage de ne pas nécessiter d'installation d'un quelconque programme pour profiter de notre approche. De plus, une fois déployé sur internet, celui-ci est accessible à quiconque souhaite améliorer la navigabilité d'une page.

Cet outil permet, d'une part, d'évaluer la navigabilité d'une page Web et d'autre part, de suggérer des recommandations pour améliorer la navigabilité de la page. L'évaluation et la suggestion de recommandations ont été implémentées en respect avec les notions présentées dans les chapitres précédents.

Implémentation du processus d'évaluation

Cette première partie est responsable de l'extraction de métriques et de l'évaluation de la navigabilité d'une page.

Notre outil est capable d'extraire les métriques d'une page Web semi-automatiquement à partir d'une adresse Internet (URL), comme le montre la page représentée par la figure 7.1. En effet, celui-ci récupère le code source nécessaire pour l'évaluation de la page Web pointée par l'URL. Comme nous le verrons dans la partie traitant l'automatisation, une confirmation des valeurs de métriques est indispensable.

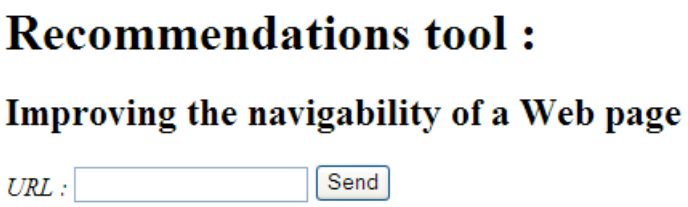


FIGURE 7.1 – Introduction d'une URL via l'outil de recommandations.

La fin du processus d'évaluation se termine en affichant le score de navigabilité de la page. L'utilisateur a dès lors le choix d'améliorer ce score, s'il ne le considère pas comme suffisant. La figure 7.2 montre cette étape pour la page http ://chronicle.com/jobs/.

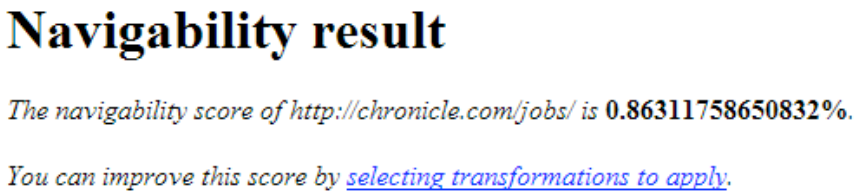


FIGURE 7.2 – Navigabilité de la page http ://chronicle.com/jobs/ calculée par l'outil de recommandations.

Implémentation du processus de recommandations

Après avoir évalué la navigabilité de la page, notre outil permet de suggérer des recommandations pour améliorer sa navigabilité.

L'interface invite alors l'utilisateur à sélectionner les transformations qu'il désire appliquer sur la page parmi les transformations possibles T_{Page}. La figure 7.3 montre cette étape de sélection. Ce choix doit être motivé en fonction des ressources disponibles. En effet, les transformations sélectionnées vont être utilisées pour suggérer des recommandations. Il est également possible de modifier les niveaux d'effort d'implémentation si ceux-ci ne correspondent pas aux coûts estimés par les experts. Cette étape se termine en affichant les meilleures séquences de transformations sélectionnées par l'algorithme heuristique.

Langage de programmation et serveur Web

Pour des raisons de facilité, nous avons choisi d'utiliser le Java comme langage de programmation. Notre application est hébergée sur un serveur Apache Tomcat qui permet, grâce aux technologies JavaServer et Java Servlet, de créer un site Web par l'intermédiaire du Java. Le choix de Tomcat comme serveur est justifié car celui-ci est fortement utilisé et connu des développeurs Web Java.

7.1.2 Automatisation de l'approche de recommandations

Toute approche de recommandations devrait être automatisée. En effet, suggérer des améliorations ne doit pas demander plus d'effort que l'implémentation de celles-ci. Automatiser l'ensemble de notre approche est cependant difficilement réalisable.

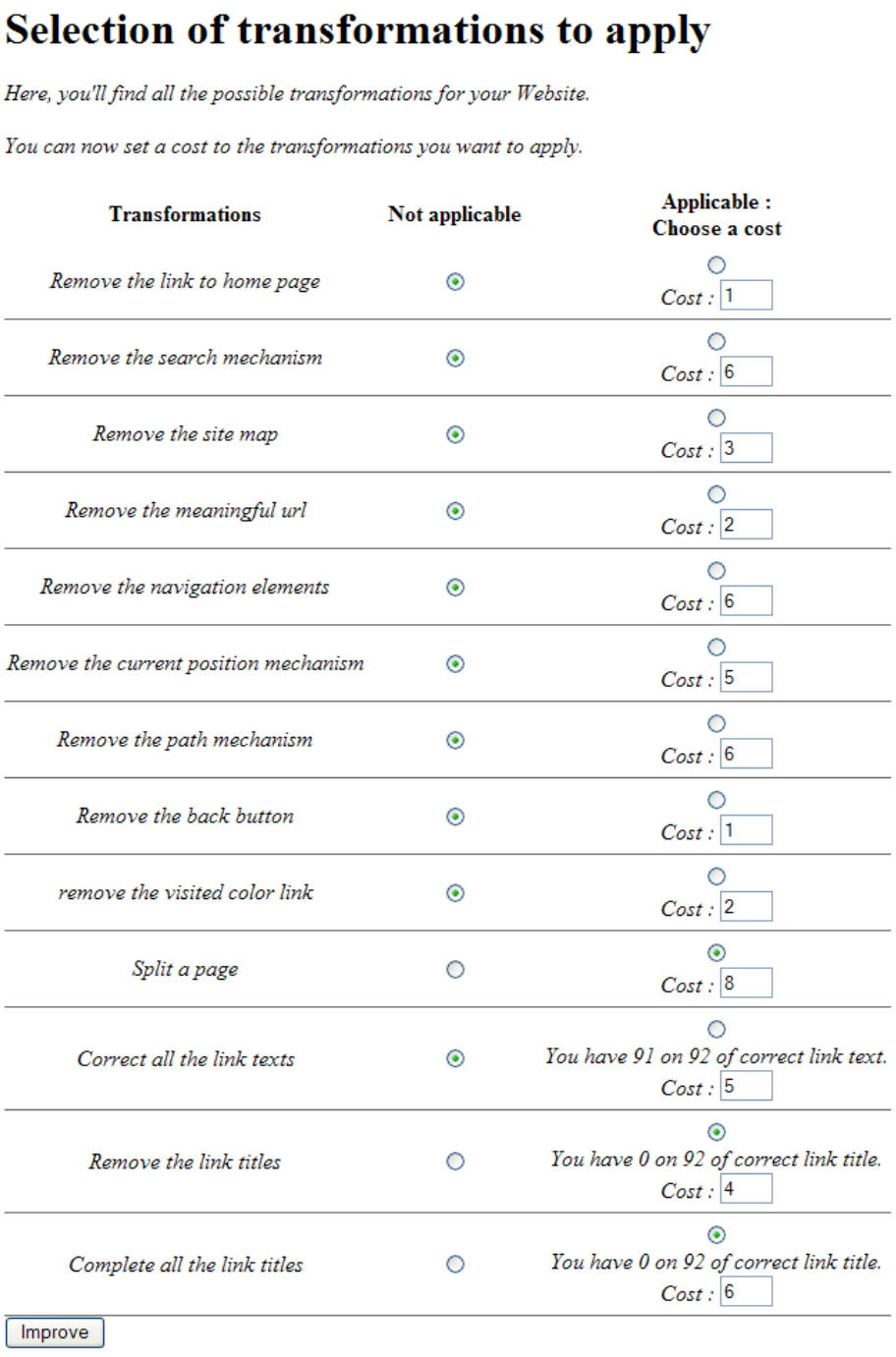
Le principal problème qui rend l'automatisation difficile se situe au niveau de l'extraction des métriques car il ne suffit pas d'analyser le code HTML d'une page Web pour en extraire l'ensemble des métriques nécessaires à l'évaluation de la qualité. Une intervention humaine est donc indispensable pour l'extraction et l'identification de certaines métriques. Ces deux difficultés sont expliquées ci-dessous.

Extraction automatique des métriques

Il est, d'une part, difficile d'extraire le code source de la page Web telle qu'elle est perçue à travers le navigateur. En effet, le contenu de nombreuses applications Web est généré dynamiquement, notamment par l'intermédiaire de langages de scripts. Par exemple, certaines applications utilisent le principe AJAX combinant du XML et Javascript asynchrone pour créer le contenu. Il est donc nécessaire d'extraire les métriques après le chargement complet de la page en interprétant le Javascript responsable de la génération.

Pour pallier à ce problème d'extraction, nous avons utilité l'API HtmlUnit[1] qui est capable de simuler le comportement d'un navigateur Web pour les pro-

[1] http ://htmlunit.sourceforge.net/



Selection of transformations to apply

Here, you'll find all the possible transformations for your Website.

You can now set a cost to the transformations you want to apply.

Transformations	Not applicable	Applicable : Choose a cost
Remove the link to home page	◉	○ *Cost :* 1
Remove the search mechanism	◉	○ *Cost :* 6
Remove the site map	◉	○ *Cost :* 3
Remove the meaningful url	◉	○ *Cost :* 2
Remove the navigation elements	◉	○ *Cost :* 6
Remove the current position mechanism	◉	○ *Cost :* 5
Remove the path mechanism	◉	○ *Cost :* 6
Remove the back button	◉	○ *Cost :* 1
remove the visited color link	◉	○ *Cost :* 2
Split a page	○	◉ *Cost :* 8
Correct all the link texts	◉	○ *You have 91 on 92 of correct link text.* *Cost :* 5
Remove the link titles	○	◉ *You have 0 on 92 of correct link title.* *Cost :* 4
Complete all the link titles	○	◉ *You have 0 on 92 of correct link title.* *Cost :* 6

Improve

FIGURE 7.3 – Sélection des transformations par l'outil de recommandations.

grammes écrits en Java. Cette API permet de naviguer dans le contenu d'une page Web telle qu'elle est perçue à travers un navigateur Web en interprétant les langages de scripts. HtmlUnit a donc été utilisée pour extraire l'ensemble des métriques nécessaires à l'évaluation de la qualité.

L'extraction des métriques est également soumise à d'autres contraintes. En effet, il est difficilement possible d'extraire le contenu d'applications Web réalisées en Flash[2], Silverlight[3] ou autres types de technologies similaires. En effet, le code source d'une page utilisant ces technologies est inaccessible étant donné qu'il est compilé. Il existe cependant certaines techniques pour en extraire visuellement le contenu, mais nous n'en parlons pas dans le cadre de ce mémoire.

Identification assistée des métriques

Le seconde difficulté est l'identification de certaines métriques comme par exemple, la présence d'un menu. Il est, en effet, particulièrement difficile de systématiser cette identification sans intervention humaine, ce qui rend l'automatisation difficile. Par exemple, bien qu'il existe des recommandations pour créer un menu, celles-ci ne sont pas nécessairement respectées. Un menu peut donc être réalisé de multiples façons. De plus, aucune balise HTML spécifique n'existe pour créer et identifier un menu.

D'autres métriques souffrent également de ce problème d'identification. Mais pour certaines d'entre elles, ce problème peut être partiellement résolu par l'intermédiaire de méthodes heuristiques. Cependant, une vérification reste nécessaire.

Certaines métriques sont, par contre, facilement identifiables et automatisables comme le nombre de liens dans une page web. Il suffit, en effet, de compter le nombre de balises HTML « a » dans la page.

Le tableau 7.1 synthétise l'ensemble des métriques nécessaires pour notre approche, indique si celles-ci sont extraites automatiquement et de quelle manière elles le sont.

Les méthodes d'extractions heuristiques sont utilisées pour faciliter la tâche d'identification de certaines métriques mais nécessitent une confirmation de la part de l'utilisateur (voir figure 7.4). Les méthodes d'extractions manuelles signifient qu'il n'existe pas de méthodes automatiques permettant l'extraction. L'utilisateur doit donc remplir manuellement des valeurs des métriques dans

[2]http ://www.adobe.com/products/flashplayer/
[3]http ://silverlight.net/

Tableau 7.1 – Tableau détaillant les métriques nécessaires pour l'évaluation de la navigabilité et leurs méthodes d'extraction

Métriques	Automatique	Méthode d'extraction
Présence d'un mécanisme de recherche	partiellement	heuristique
Présence d'un plan de site	partiellement	heuristique
Présence d'un URL relatif	partiellement	heuristique
Présence d'un lien vers la page d'accueil	partiellement	heuristique
Présence d'un menu	non	manuelle
Présence d'un indicateur de la position courante	non	manuelle
Présence d'un indicateur de chemin	non	manuelle
Présence d'un changement de couleur du lien visité	non	manuelle
Présence d'un bouton Retour Actif	non	manuelle
Nombre de liens par page	oui	comptage
Temps de téléchargement de la page	oui	mesure
Pourcentage de textes de liens significatifs	oui	comptage
Pourcentage de titres de liens	oui	comptage

le formulaire prévu à cet effet comme le montre également la figure 7.4.

Les autres méthodes d'extraction nécessitent un comptage ou une mesure. Pour l'extraction de la valeur de métriques du pourcentage de textes de liens significatifs, l'outil utilise les mots « learn more », « more », « previous », « click », « click here », « next », « even more », « all », « read more », « see more », « read », « here », « continued », « see here », « see », « find out more », « plus » pour calculer le pourcentage de liens représentatifs. Pour les titres de liens et les nombres de liens, l'outil se charge de les compter. La valeur de métrique qui concerne le temps de téléchargement est réalisé en calculant en millisecondes le temps nécessaire pour charger l'entièreté de la page.

L'interface Web avec le formulaire demandant une confirmation (pour les métriques heuristiques) et une introduction (pour les métriques manuelles) des valeurs des métriques est représentée par la figure 7.4. Cette figure montre cette étape d'extraction pour la page http ://chronicle.com/jobs/.

FIGURE 7.4 – Formulaire de confirmations de d'introductions de valeurs de métriques pour l'outil de recommandations.

7.1.3 Design de l'outil

Design patern et implantation modulaire

L'outil de recommandations respecte certains concepts de programmation. En effet, l'implémentation se veut modulaire et représente chacun des concepts abordés de ce mémoire comme la notion de transformation, de solution, d'algorithme, ... Ceci permet une meilleure compréhension du code source de notre outil.

De plus, le design du code est réalisé de façon à ce que chacun des paramètres et ensemble de transformations soient facilement modifiables. Le design patern statégie a par ailleurs été utilisé pour le caclul de la fonction

objective, ce qui permet l'ajout de nouvelles fonctions objectives sans devoir réaliser des modifications dans le code.

Implémentation du réseau Bayésien

Nous avons donc utilisé BNJ (Bayesian Java Network [4]), une API open source écrite en Java pour implémenter le réseau de navigabilité bayésien. Celle-ci, développée par KDD Lab dans l'Université de l'état du Kansas aux USA, a été privilégiée notamment pour sa facilité d'utilisation. Notre outil utilise la dernière version de BNJ disponible depuis mai 2006. Bien que son développement se soit arrêté, les fonctionnalités implémentées étaient suffisantes pour notre approche de recommandations. BNJ dispose d'une interface graphique permettant de manipuler les réseaux bayésiens, mais celle-ci n'a pas employée dans le cadre de cette thèse.

Vous trouverez en annexe un comparatif reprenant différentes solutions permettant de modéliser et faire de l'inférence avec les réseaux bayésiens.

7.2 Étude pratique

7.2.1 Objectifs de l'expérimentation

Le but principal de cette étude est de vérifier la faisabilité et la pertinence de notre approche de recommandations. En effet, celle-ci doit pouvoir s'inscrire dans une démarche d'amélioration de la qualité d'une application Web. Un développeur doit pouvoir utiliser notre outil pour faire face aux problèmes de qualité de son application.

Nous avons donc évalué notre approche selon deux objectifs :

- Vérifier que l'algorithme méta-heuristique trouve bien la solution optimale. Dans le cas contraire, déterminer la différence entre la solution trouvée avec la solution optimale.
- Mesurer si le temps d'exécution pour suggérer des recommandations est raisonnable, étant donné que la taille de l'espace de recherche est exponentiel par rapport aux nombres de transformations. Ce critère est également important pour l'évolutivité de notre approche. En effet, dans le cadre de ce mémoire, nous évaluons la navigabilité d'une page Web mais il devrait être possible d'évaluer la navigabilité de toute une application. De plus, notre approche se voulant générale, il est donc possible de l'adapter à d'autres MQ et de définir d'autres ensembles de transformations $t_{candidates}$, potentiellement plus grands que T_{NAV}. Ces évolutions

[4] http ://bnj.sourceforge.net/

peuvent donc potentiellement augmenter la taille de l'espace de recherche et donc le temps de calcul.

7.2.2 Déroulement de l'expérimentation

Nous avons testé notre outil sur 15 pages Web. Certaines pages ont été choisies aléatoirement tandis que d'autres ont été sélectionnées parmi la liste des meilleurs sites Web répertoriés sur http ://www.webbyawards.com et parmi les pires sites Web répertoriés sur http ://www.worstoftheweb.com/.

Ces 15 pages ont volontairement une navigabilité différente (q_{NAV}) et un espace de recherche différent (Espace) (cfr. chapitre 6). L'espace de recherche varie en fonction du nombre de transformations à considérer (T_{Page}). Comme nous l'avons expliqué dans la section ≪ Sélection des meilleures transformations ≫ du chapitre 6, certaines transformations ont volontairement été enlevées pour diminuer l'espace de recherche de l'algorithme heuristique et donc améliorer les performances. Ceci justifie un nombre différent pour chaque page Web.

Le tableau 7.3 détaille l'ensemble des 15 pages Web avec leurs niveaux de navigabilité et la taille de leurs espaces de recherche. On se rend compte que les pages Web avec un faible nombre de transformations ont généralement un meilleur score de navigabilité. En effet, ceci s'explique car les pages possédant un bon score de navigabilité ont besoin de considérer moins de transformations pour améliorer leurs scores.

7.2.3 Paramètres de l'expérimentation

Paramètre de l'algorithme heuristique

L'algorithme du recuit simulé responsable de la recherche de la meilleure solution nécessite l'ajustement de certains paramètres, ceux-ci dépendant du problème à traiter. Dans le cadre de ce mémoire, nous avons ajusté la valeur des paramètres à notre problème de sélection en suivant les recommandations issues de la littérature comme présenté dans le chapitre 3 :

La température initiale (Trs_0) doit être suffisamment grande pour permettre virtuellement que toutes les transitions soient acceptées. Selon [2], ceci est possible si le taux d'acceptation initial ($e^{\Delta Obj/Trs}$) est proche de 1. Pour notre problème de recommandation, nous avons choisi une température initiale de 10 car elle donne un taux d'acceptation initial supérieur à 95,12%.

Ce nombre est obtenu en calculant la valeur du taux d'acceptation pour la pire des solutions potentielles, c'est-à-dire une solution dont le coût est

Tableau 7.2 – Navigabilité initiale et taille de l'espace de recherche des 15 pages Web étudiées.

#	Page	q_{NAV}	Espace
1	http ://www.zombietime.com/vi_day/	13 %	21
2	http ://www.consc.net/chalmers/	19%	17
3	http ://catsinsinks.com/	24%	17
4	http ://www.graphics.cornell.edu/people/director.html	25%	15
5	http ://www.lendingclub.com/home.action	68%	13
6	http ://www.hotelgreens.co.uk/	69%	13
7	http ://www.outbackphoto.com/index_news.html	73%	15
8	http ://www.cio.com/ ?source=top_nav	73%	15
9	http ://community.oreilly.com/	76%	9
10	http ://www.loveisrespect.org/is-this-abuse/index.html	77%	11
11	http ://www.networkcomputing.com	83%	13
12	http ://wiki.sidekick.com/ ?t=anon	84%	11
13	http ://shopping.aol.com/pages/Holiday-Hot-List/9191	84%	9
14	http ://chronicle.com/jobs/	84%	11
15	http ://www.cnn.com/2008/WORLD/africa/12/10/zim-babwe.epidemic.cholera.spread/index.html	85%	9

supérieur à la contrainte de coût W et que la différence de navigabilité est de 100%. Ceci a été réalisé en utilisant les formules du chapitre précédent.

Mathématiquement, $e^{\Delta Obj/10} = 95,12\%$ avec $\Delta Obj = -0,5$. La valeur de ΔObj est obtenue en utilisant la fonction objective pour la solution initiale ($Obj(s_0) = 0,5$) et la pire solution ($Obj(s_{pire}) = 0$), c'est-à-dire $\Delta Obj := Obj(s_{pire}) - Obj(s_0) = 0 - 0,5 = -0,5$ selon l'algorithme 2 du recuit simulé. Où $Obj(s_{pire}) = Amé\_Nav * 0.5 = 0$, c'est-à-dire pour une différence de navigabilité étant de 100%, $Amé\_Nav = (1 + (-100\%))/2 = 0$. Et, $Obj(s_0) = 0.5 + Amé\_Nav * 0.5 = 0,5$

Le paramètre α permettant de décroître le paramètre de contrôle doit être compris entre $0,8$ et $0,99$ [2]. Vu le faible nombre de transformations à considérer pour notre approche de recommandations, l'espace de recherche reste relativement petit. Nous avons donc choisi $\alpha = 0,8$.

La longueur de la température (Nrs) est fixée dans le but d'atteindre un certain équilibre. Cette valeur étant généralement déduite après un certain nombre d'expériences réalisées au préalable a été fixée à $Nrs = 30$.

Le critère d'arrêt permettant d'arrêter la température lorsque qu'elle s'approche de 0 a été fixé à $Trs \leq 0.001$.

Variables des transformations

Dans la chapitre 6, nous avons présenté l'ensemble des transformations T_{NAV}. Les transformations « Ajout/Suppression d'un menu », « Ajout/Suppression d'un indicateur de chemin » et « Diviser/fusionner une page » ont un effet sur un nombre variable de liens. Nous avons fixé ces variables empiriquement. Nous considérons donc qu'un menu est environ composé de 6 liens : $b^* = 6$. Dans le cadre de nos tests, un indicateur de chemin contient en moyenne 2 liens : $c^* = 2$. Pour diviser ou fusionner une page, la valeur de la variable d^* est fixée en fonction de la présence d'un indicateur de chemin et de la présence d'un menu en utilisant les valeurs b^* et c^*.

La transformation responsable de diviser ou fusionner une page est soumise à une pré-condition limitant le nombre de division. Le nombre minimal de liens dans un page doit être supérieur à 29. Ce nombre correspond au nombre maximal de liens possibles pour avoir le pourcentage maximal de la classe floue « Petit » du noeud d'entrée représentant le nombre de liens par page.

Déroulement de l'expérimentation : Choix de contraintes de coût

Notre approche de recommandation tente de sélectionner la meilleure séquence de transformations pour une limite de coût W, comme le montre l'équation 6.1. Pour faciliter le choix de cette contrainte W, nous avons fixé 4 niveaux de coût. Ceux-ci sont définis par rapport au coût total d'une solution W_{MAX} calculé en additionnant l'ensemble des coûts des transformations sélectionnées pour la suggestion de recommandations. W_{MAX} correspond donc à l'effort nécessaire pour appliquer l'ensemble des transformations sélectionnées.

Les 4 contraintes de coût pour l'ensemble de nos expérimentations sont : $< 0,25W_{MAX}$, $< 0,5W_{MAX}$, $< 0,75W_{MAX}$ et $< W_{MAX}$.

7.2.4 Résultats de l'expérimentation

Pour chacune des pages considérées et pour chaque contrainte de coût, l'algorithme du recuit simulé à trouvé la meilleure solution possible. Nous en déduisons que notre premier objectif est atteint. Ce résultat n'est pas surprenant. En effet, les paramètres de l'algorithme permettant la sélection ont été adaptés dans le but d'atteindre cet objectif. Cependant, pour que notre approche soit cohérente, il faut que cette sélection soit réalisée dans un temps raisonnable (voir ci-après).

Les résultats (les scores de navigabilité) de notre expérimentation pour chacun des paliers de coût est illustré dans le tableau 7.3.

Tableau 7.3 – Scores de navigabilité des pages Web étudiées pour chaque contrainte de coût

Page	0%	25%	50%	75%	100%
1	0,13	0,85	0,86	0,91	0,92
2	0,19	0,77	0,88	0,91	0,92
3	0,24	0,69	0,88	0,91	0,92
4	0,25	0,69	0,88	0,91	0,92
5	0,68	0,77	0,89	0,90	0,92
6	0,69	0,79	0,88	0,91	0,92
7	0,73	0,86	0,88	0,91	0,92
8	0,73	0,86	0,87	0,91	0,92
9	0,76	0,86	0,87	0,90	0,92
10	0,77	0,78	0,88	0,91	0,92
11	0,83	0,86	0,87	0,91	0,92
12	0,84	0,86	0,89	0,91	0,92
13	0,84	0,86	0,86	0,90	0,91
14	0,84	0,86	0,88	0,91	0,92
15	0,85	0,86	0,89	0,91	0,92

Pour des raisons de visibilité, nous avons synthétisé ces résultats en 3 groupes de pages (les valeurs sont reprises dans le tableau 7.3) : les pages avec une mauvaise navigabilité ($q_{NAV} \leq 25\%$), les pages avec une bonne navigabilité et les pages avec une très bonne navigabilité ($q_{NAV} \geq 80\%$). Le premier groupe reprend les pages numérotées de 1 à 4, le second les pages de 5 à 10 et le dernier les pages de 11 à 15.

L'amélioration de navigabilité après la première contrainte de coût (25% du coût total) est très significative pour les pages avec une mauvaise navigabilité. Ces pages augmentent leurs niveaux de navigabilité d'environ 50% pour atteindre un score d'environ 75%. La différence est moins nette pour les pages de bonne navigabilité (environ 10%) et quasi inexistante pour le troisième groupe. La raison de cette différence est simple. L'algorithme identifie et choisit les transformations les plus efficientes en premier. Les pages avec un mauvais niveau de navigabilité ont donc nécessairement une amélioration plus importante que celles qui possèdent déjà un bon niveau. En effet, peu de changements est nécessaire pour améliorer la qualité des pages déjà navigables. Nous pouvons cependant signaler que notre approche de recommandations est particulièrement efficace pour la page peu navigable.

Une autre explication vient du nombre de transformations à considérer (T_{Page}) par l'algorithme. En effet, les pages avec un mauvais niveau de na-

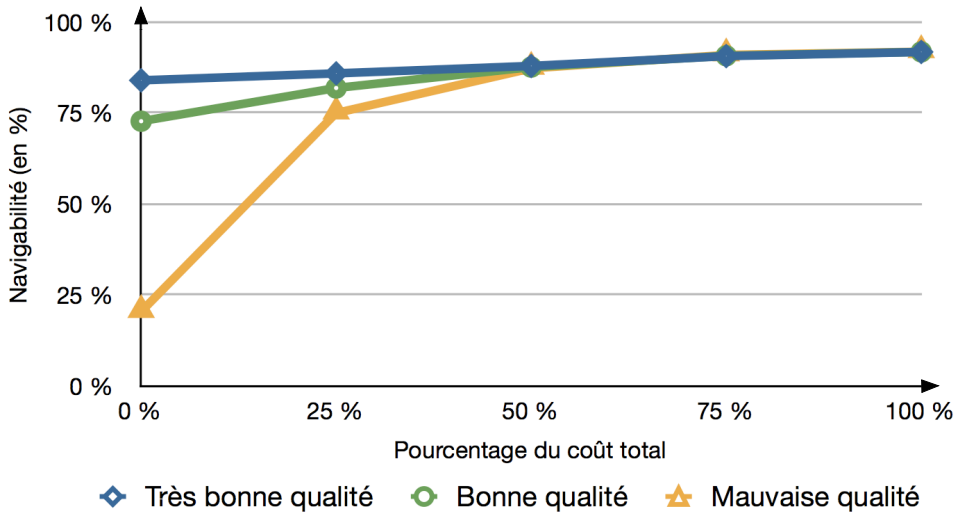


FIGURE 7.5 – Score de navigabilité par groupes de pages pour chacun des paliers de coût

vigabilité ont un nombre de transformations plus important. Leur coût total W_{MAX} est par conséquent plus élevé que pour les pages de meilleure qualité. Un coût total plus élevé permet de considérer plus de transformations par pallier de coût et donc un nombre plus important de changements améliorant le niveau de navigabilité, ce qui justifie un plus grand écart entre ces pages.

Pour la deuxième et troisième contraintes de coût, la différence de navigabilité est beaucoup moins importante. Ce qui est logique vu les remarques énoncées ci-dessus. Les trois groupes de pages finissent par converger vers le meilleur niveau de navigabilité possible qui se situe près des 92%. Celui-ci est atteint lorsqu'on atteint la limite de coût ($W = W_{MAX}$).

La figure 7.6 montre la différence de temps d'exécution entre l'algorithme du recuit simulé utilisé pour notre outil de recommandations et un algorithme exhaustif en fonction du nombre de transformations considérées. Ce graphique a été réalisé en faisant la moyenne des temps pour un même nombre de transformations.

L'algorithme exhaustif utilisé sélectionne la meilleure séquence de transformation en considérant toujours le même ordre dans la séquence, ce qui simplifie le problème. Le graphique 7.6 montre que même avec cette simplification qui engendre un temps de calcul plus rapide, l'algorithme heuristique est largement plus rapide que l'algorithme exhaustif quand le nombre de transformations augmente et cette différence croît de façon exponentielle.

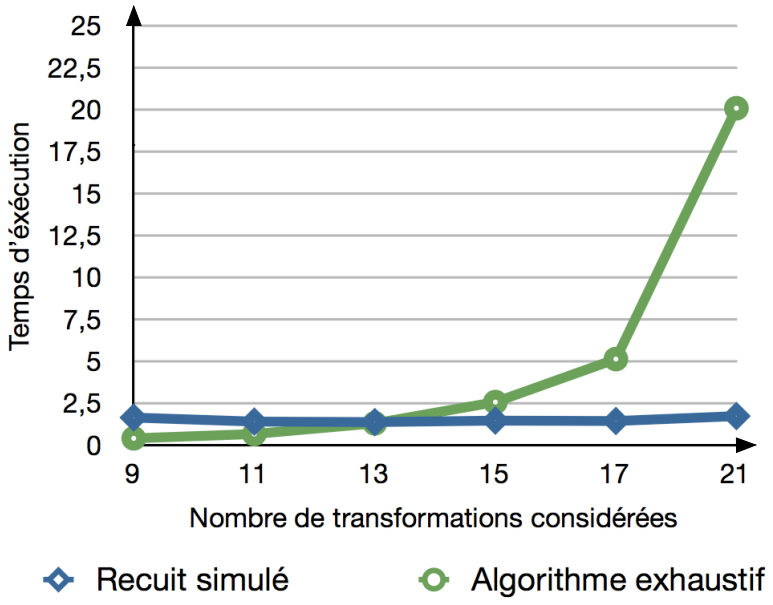


FIGURE 7.6 – Comparaison des temps d'exécution en fonction du type d'algorithme et du nombre de transformations considérées T_{Page}

Le graphique montre également que l'algorithme du recuit simulé sélectionne la meilleure séquence dans un temps constant (de l'ordre de la seconde) même si le nombre de transformations augmente. En effet, des paramètres identiques pour l'algorithme heuristique ont été utilisés pour chacune des pages. Par conséquent, l'algorithme effectue toujours le même nombre d'itération, ce qui justifie ce temps constant.

Ces résultats sont donc à interpréter minutieusement (le temps d'exécution de l'algorithme dépend directement de ses paramètres). En effet, les paramètres doivent être adaptés en fonction du problème à traiter et peuvent donc être modifiés si le problème évolue (évaluation de l'entièreté d'une application Web, utilisation d'un autre modèle de qualité et/ou considération d'un autre ensemble de transformations).

Nous pouvons cependant affirmer que notre deuxième objectif est atteint. En effet, notre algorithme sélectionne dans tous les cas la meilleure séquence de transformations (même si le nombre de transformations à considérer est plus élévé) dans un temps raisonnable constant.

Chapitre 8

Conclusion

8.1 Contributions

Ce mémoire présente une approche générale de recommandations pour améliorer le niveau de qualité des applications Web. Cette approche s'inspire de travaux réalisés dans le domaine de l'évaluation et l'amélioration de la qualité (du logiciel et du Web) et a pour objectif de combler les manques dans le domaine du Web.

Pour améliorer le niveau qualité d'une application Web avec un effort limité, ce mémoire se base sur un modèle de qualité (MQ) probabiliste, un ensemble de transformations candidates et une estimation de l'effort d'implémentation pour proposer une méthode permettant de sélectionner la meilleure séquence de transformations à implémenter sur l'application donnée.

Notre approche consiste en deux processus : un processus d'évaluation utilisant un MQ probabiliste comme boîte noire pour évaluer les effets des transformations sur la qualité de l'application et un processus de recommandations qui utilise un algorithme heuristique pour proposer la séquence de transformations optimales pour améliorer la qualité. L'utilisation d'un MQ probabiliste se justifie, car celui-ci permet de pallier les problèmes d'incertitudes (subjectivité, imprécision,...) inhérents à l'évaluation de la qualité. Notre approche de recommandations, de complexité exponentielle par rapport au nombre de transformations, justifie l'utilisation d'un algorithme heuristique permettant l'exploration d'un grand espace de solutions dans un temps raisonnable.

Nous illustrons notre approche de recommandations sur une caractéristique de la qualité, la navigabilité. Pour ce faire, nous utilisons un MQ existant évaluant la navigabilité d'une page Web et nous définissons concrètement un ensemble de transformations qui influencent le niveau de navigabilité d'une page Web ainsi que ses effets sur le MQ. Nous évaluons ensuite la faisabilité de notre approche en l'appliquant sur 15 pages Web. Pour ce faire,

nous avons développé un outil basé sur les technologies Web, responsable d'évaluer et suggérer des améliorations pour la navigabilité d'une page Web. Notre expérimentation permet de montrer que notre algorithme heuristique sélectionne dans un temps constant la meilleure séquence de transformations pour les 15 pages. Nous pouvons également supposer que notre approche résistera bien à l'évolutivité du problème, c'est-à-dire à l'évaluation de la qualité d'une application Web et à et l'utilisation d'autres MQ.

Notre contribution pour l'évaluation et l'amélioration de la qualité illustrée à travers ce mémoire peut se résumer de la façon suivante :

- une approche générale d'évaluation et d'amélioration de la qualité d'une application Web ;
- un lien explicite entre l'évaluation et l'amélioration de la qualité ;
- une prise en compte de l'effort d'implémentation des transformations dans le processus de recommandation ;
- une utilisation d'un MQ pour fournir une estimation précise du niveau de qualité d'une application Web selon un ensemble de règles et de standards ;
- une évaluation de la qualité probabiliste pour traiter les problèmes d'incertitudes ;
- une utilisation d'un algorithme heuristique pour suggérer des recommandations et pallier au défi de complexité en temps ;
- un outil capable d'évaluer et de suggérer des recommandations pour améliorer la navigabilité d'une page.

8.2 Limites et travaux futurs

Ce mémoire illustre une approche générale permettant de suggérer des recommandations pour améliorer la qualité des applications Web en utilisant un MQ. Celle-ci se veut adaptable et extensible à tout type de MQ. Dans le cadre de ce mémoire, nous l'avons illustrée pour évaluer et améliorer la navigabilité d'une page Web.

Il serait intéressant d'adapter notre approche, dans des travaux futurs, à d'autres MQ évaluant d'autres caractéristiques de la qualité, comme l'utilisabilité par exemple. Ceci nécessiterait dès lors définir d'autres ensembles de transformations qui influencent ces niveaux de qualité. Notre approche pourrait également être adaptée pour améliorer la qualité de toute une application.

De plus, il serait également intéressant de valider notre approche auprès d'utilisateurs, dans le but de savoir si les améliorations suggérées ont du sens pour eux.

Ce mémoire expérimente notre approche de recommandations sur 15 pages Web. Bien que ce nombre soit faible, il permet d'avoir une bonne idée du temps nécessaire pour proposer des suggestions. Cela nous a permis notamment de nous rendre compte que l'utilisation d'un algorithme heuristique se justifie par rapport à un algorithme exhaustif pour un nombre de transformations élevé dans le but de garder un temps de calcul raisonnable. En effet, nos expérimentations ont montré que le temps de calcul pour un algorithme exhaustif augmente exponentiellement en fonction du nombre de transformations à considérer, alors que l'algorithme heuristique trouve la meilleure séquence dans un temps constant. Tester notre approche sur un nombre plus important de pages et de transformations permettrait de confirmer cette hypothèse.

Il serait également intéressant de tester différentes sortes d'algorithmes pour sélectionner la séquence de transformations comme des algorithmes génétiques. En effet, bien que nous supposons que l'algorithme heuristique du recuit simulé reste valable pour un nombre de transformations plus grand (dépendant du nombre de transformations à évaluer), la recherche locale qu'il met en oeuvre peut-être limitée dans le cas de grands espaces de recherche. Une recherche globale ou par population, comme dans les cas des algorithmes génétiques, peut donner de meilleurs résultats.

Bibliographie

[1] World wide web consortium (w3c) to develop interoperable technologies (specifications, guidelines, software, and tools) to lead the web to its full potential (http ://www.w3.org/), dernier accès le 30 juillet 2009.

[2] Emile Aarts and Jan Korst. *Simulated annealing and Boltzmann machines : a stochastic approach to combinatorial optimization and neural computing*. John Wiley & Sons, Inc., New York, NY, USA, 1989.

[3] Albuquerque A. B. and Belchior A.D. E-commerce websites : a qualitative évaluation. In *the 11th International World Wide Web Conference (WWW)*, 2002.

[4] Cornelia Boldyreff and Paolo Tonella. Web site evolution. *Journal of Software Maintenance*, 16(1-2) :1–4, 2004.

[5] Salah Bouktif. *Amélioration de la prédiction de la qualité du logiciel par combinaison et adaptation de modèles*. PhD thesis, Université de Montréal, 2005.

[6] G. Brajnik. Automatic web usability evaluation : Where is the limit ? In *the 6th Conference on Human Factors & the Web*, 2000.

[7] Boldyreff C., Gaskell C., Marshall A., and Warren P. Web-sem project : Establishing effective web site evaluation metrics. In *the 2nd International Workshop on Web Site Evolution (WSE)*, 2000.

[8] Coral Calero, Angélica Caro, and Mario Piattini. An applicable data quality model for web portal data consumers. *World Wide Web*, 11(4) :465–484, 2008.

[9] Gregory F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks (research note). *Artif. Intell.*, 42(2-3) :393–405, 1990.

[10] S. Dalton. *A Workbench to Support Development and Maintenance of World Wide Web Documents*. PhD thesis, Department of Computer Science, University of Durham, 1996.

[11] Christophe Deleuze. Some points affecting web performance. In *Computer Networks, Vol. 50, No 10*, pages 1533–1546, 2006.

[12] Norman Fenton, Paul Krause, and Martin Neil. Software measurement : Uncertainty and causal modeling. *IEEE Software*, 19(4) :116–122, 2002.

[13] Norman Fenton and Martin Neil. Making decisions : Using bayesian nets and mcda. *Knowledge-Based Systems*, 14 :307–325, 2000.

[14] Norman Fenton, Martin Neil, and David Marquez. Using bayesian networks to predict software defects and reliability. *Journal of Risk and Reliability*, 2008.

[15] Jacques A. Ferland. Heuristic search methods for combinatiorial programming problems. Technical report, Université de Montréal, 2001.

[16] Alejandra Garrido, Gustavo Rossi, and Damiano Distante. Model refactoring in web applications. In *WSE '07 : Proceedings of the 2007 9th IEEE International Workshop on Web Site Evolution*, pages 89–96, Washington, DC, USA, 2007. IEEE Computer Society.

[17] Mark Harman and Laurence Tratt. Pareto optimal search based refactoring at the design level. In *GECCO '07 : Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1106–1113, New York, NY, USA, 2007. ACM.

[18] Jim Isaak. Web site engineering best practice standards (ieee 2001). In *WSE '02 : Proceedings of the Fourth International Workshop on Web Site Evolution (WSE'02)*, page 81, Washington, DC, USA, 2002. IEEE Computer Society.

[19] Ghazwa Malak. *Évaluation de la Qualité des Applications Web : Approche ProbabilisteApproche Probabiliste*. PhD thesis, Département d'Informatique et de Recherches Opérationnelles, Université de Montréal, 2007.

[20] V.K. Namasivayam, V.K. ; Prasanna. Scalable parallel implementation of exact inference in bayesian networks. In *12th International Conference on Parallel and Distributed Systems (ICPADS)*, 2006.

[21] Jakob Nielsen and Hoa Loranger. *Prioritizing web usability*. New Riders, Berkeley, Calif., 2006.

[22] Luis Olsina and Gustavo Rossi. Measuring web application quality with webqem. *IEEE MultiMedia*, 9(4) :20–29, 2002.

[23] Luis Olsina, Gustavo Rossi, Alejandra Garrido, Damiano Distante, and Gerardo Canfora. Incremental quality improvement in web applications using web model refactoring. In *WISE Workshops*, pages 411–422, 2007.

[24] Yu Ping and Kostas Kontogiannis. Refactoring web sites to the controller-centric architecture. In *CSMR '04 : Proceedings of the Eighth Euromicro Working Conference on Software Maintenance and Reengineering (CSMR'04)*, page 204, Washington, DC, USA, 2004. IEEE Computer Society.

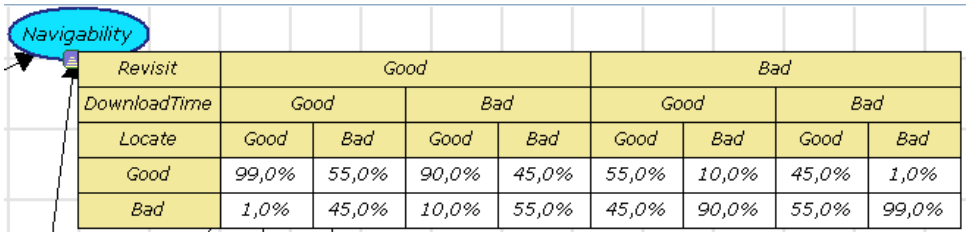
[25] Houari A. Sahraoui, Robert Godin, and Thierry Miceli. Can metrics help to bridge the gap between the improvement of oo design quality and its automation ? *Software Maintenance, IEEE International Conference on*, 0 :154, 2000.

[26] Olaf Seng, Johannes Stammel, and David Burkhart. Search-based determination of refactorings for improving the class structure of object-oriented systems. In *GECCO '06 : Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1909–1916, New York, NY, USA, 2006. ACM.

[27] W3C. Web content accessibility guidelines 1.0 : W3c recommendation 5-may-1999 (http ://www.w3.org/tr/wai-webcontent/), dernier accès le 30 juillet 2009.

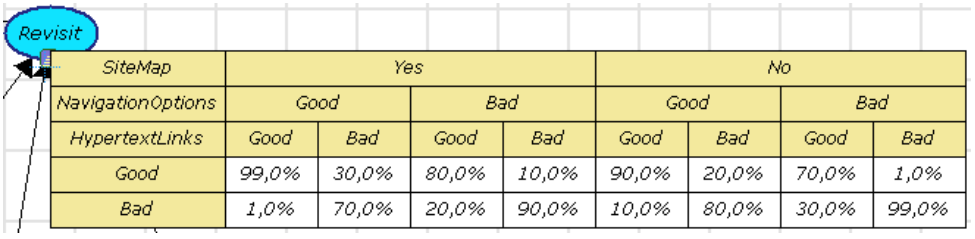
Annexes

8.3 Tables de probabilités des noeuds du réseau bayésien évaluant la navigabilité d'une page Web



Revisit	Good				Bad			
DownloadTime	Good		Bad		Good		Bad	
Locate	Good	Bad	Good	Bad	Good	Bad	Good	Bad
Good	99,0%	55,0%	90,0%	45,0%	55,0%	10,0%	45,0%	1,0%
Bad	1,0%	45,0%	10,0%	55,0%	45,0%	90,0%	55,0%	99,0%

FIGURE 8.1 – Table de probabilités d'un noeud intermédiaire du modèle de navigabilité bayésien.



SiteMap	Yes				No			
NavigationOptions	Good		Bad		Good		Bad	
HypertextLinks	Good	Bad	Good	Bad	Good	Bad	Good	Bad
Good	99,0%	30,0%	80,0%	10,0%	90,0%	20,0%	70,0%	1,0%
Bad	1,0%	70,0%	20,0%	90,0%	10,0%	80,0%	30,0%	99,0%

FIGURE 8.2 – Table de probabilités d'un noeud intermédiaire du modèle de navigabilité bayésien.

Locate

NavigationMechanisms	Good		Bad	
UserFeedback	Good	Bad	Good	Bad
Good	99,0%	80,0%	20,0%	1,0%
Bad	1,0%	20,0%	80,0%	99,0%

FIGURE 8.3 – Table de probabilités d'un noeud intermédiaire du modèle de navigabilité bayésien.

NavigationOptions

LinkToHome	Yes		No	
BackButton	Yes	No	Yes	No
Good	99,0%	70,0%	30,0%	1,0%
Bad	1,0%	30,0%	70,0%	99,0%

FIGURE 8.4 – Table de probabilités d'un noeud intermédiaire du modèle de navigabilité bayésien.

HypertextLinks

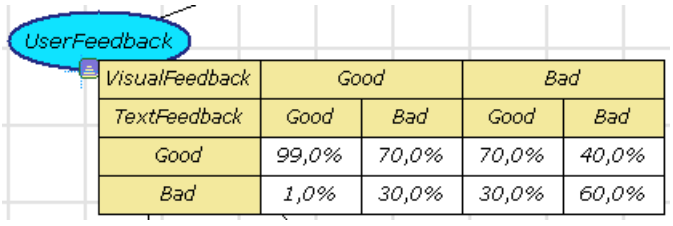
NavigationElements	Yes						No					
pathMechanism	Yes			No			Yes			No		
LinksNumber	High	Medium	Low	High	Medium	Low	High	Medium	Low	High	Medium	Low
Good	80,0%	90,0%	99,0%	80,0%	85,0%	90,0%	10,0%	15,0%	20,0%	1,0%	5,0%	10,0%
Bad	20,0%	20,0%	1,0%	20,0%	15,0%	10,0%	90,0%	85,0%	80,0%	99,0%	95,0%	90,0%

FIGURE 8.5 – Table de probabilités d'un noeud intermédiaire du modèle de navigabilité bayésien.

NavigationMechanisms

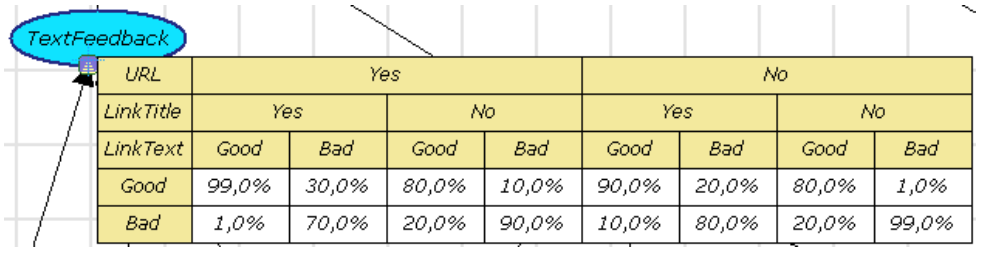
SiteMap	Yes				No			
NavigationElements	Yes		No		Yes		No	
SearchMechanism	Yes	No	Yes	No	Yes	No	Yes	No
Good	99,0%	70,0%	45,0%	15,0%	85,0%	55,0%	30,0%	1,0%
Bad	1,0%	30,0%	55,0%	85,0%	15,0%	40,0%	70,0%	99,0%

FIGURE 8.6 – Table de probabilités d'un noeud intermédiaire du modèle de navigabilité bayésien.



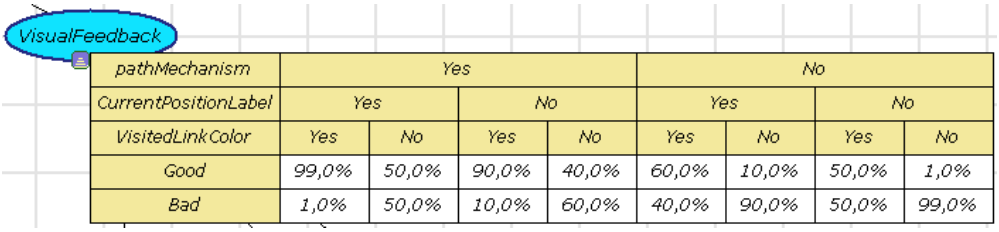
VisualFeedback	Good		Bad	
TextFeedback	Good	Bad	Good	Bad
Good	99,0%	70,0%	70,0%	40,0%
Bad	1,0%	30,0%	30,0%	60,0%

FIGURE 8.7 – Table de probabilités d'un noeud intermédiaire du modèle de navigabilité bayésien.



URL	Yes				No			
LinkTitle	Yes		No		Yes		No	
LinkText	Good	Bad	Good	Bad	Good	Bad	Good	Bad
Good	99,0%	30,0%	80,0%	10,0%	90,0%	20,0%	80,0%	1,0%
Bad	1,0%	70,0%	20,0%	90,0%	10,0%	80,0%	20,0%	99,0%

FIGURE 8.8 – Table de probabilités d'un noeud intermédiaire du modèle de navigabilité bayésien.



pathMechanism	Yes				No			
CurrentPositionLabel	Yes		No		Yes		No	
VisitedLinkColor	Yes	No	Yes	No	Yes	No	Yes	No
Good	99,0%	50,0%	90,0%	40,0%	60,0%	10,0%	50,0%	1,0%
Bad	1,0%	50,0%	10,0%	60,0%	40,0%	90,0%	50,0%	99,0%

FIGURE 8.9 – Table de probabilités d'un noeud intermédiaire du modèle de navigabilité bayésien.

8.4 Fonctions Objectives

Cette section montre des exemples de fonctions objectives types utilisées pour les expérimentations.

Exemple de fonction objective accordant un avantage de 75% à l'amélioration de navigabilité et de 25% à l'effort proportionnel $pCoût(s)$ au coût total (le pourcentage peut varier) :

$$Obj(s) = 0.75 * Amé\_Nav + 0.25 * pCoût(s)$$

où

$$pCoût(s) = \frac{coût(s)}{CoûtTotal} * 10$$

$coût(s)$ correspond à l'effort nécessaire pour implémenter la solution s.
$CoûtTotal$ est la somme des coûts des transformations sélectionnées.

Exemples de fonctions avec un seuil de navigabilité s_{NAV} accordant un avantage proportionnel aux solutions respectant la limite de ce seuil (celui-ci est à définir lors des expérimentations) :

$$Obj(s) = \begin{cases} 0.25 + \frac{q_{NAV}(s)}{1+coût(s)} * 0.75 & \text{si} \quad \Delta Nav(s_0, s) \leq s_{NAV} \\ \frac{q_{NAV}(s)}{1+coût(s)} * 0.25 & \text{si} \quad \Delta Nav(s_0, s) > s_{NAV} \end{cases}$$

$$Obj(s) = \begin{cases} 0.5 + \frac{Amé\_Nav}{1+pCoût(s)} * 0.5 & \text{si} \quad \Delta Nav(s_0, s) \leq s_{NAV} \\ \frac{Amé\_Nav}{1+pCoût(s)} * 0.5 & \text{si} \quad \Delta Nav(s_0, s) > s_{NAV} \end{cases}$$

$$Obj(s) = \begin{cases} 0.25 + \frac{\Delta Nav(s_0,s)}{1+coût(s)} * 0.75 & \text{si} \quad \Delta Nav(s_0, s) \leq s_{NAV} \\ \frac{\Delta Nav(s_0,s)}{1+coût(s)} * 0.25 & \text{si} \quad \Delta Nav(s_0, s) > s_{NAV} \end{cases}$$

$$Obj(s) = \begin{cases} 0.25 + \frac{\Delta Nav(s_0,s)}{1+coût(s)} & \text{si} \quad \Delta Nav(s_0, s) \leq s_{NAV} \\ \frac{\Delta Nav(s_0,s)}{1+coût(s)} * \sin(\frac{q_{NAV}(s)}{(2/\pi)*q_{NAV}(s)}) * 0.25 & \text{si} \quad \Delta Nav(s_0, s) > s_{NAV} \end{cases}$$

$$Obj(s) = \begin{cases} 0.25 + \frac{q_{NAV}(s)}{(1+coût(s))/(1+CoûtTotal)} * 0.75 & \text{si} \quad \Delta Nav(s_0, s) \leq s_{NAV} \\ \frac{q_{NAV}(s)}{(1+coût(s))/(1+CoûtTotal))} * 0.25 & \text{si} \quad \Delta Nav(s_0, s) > s_{NAV} \end{cases}$$

$$Obj(s) = \begin{cases} \frac{q_{NAV}(s)}{(1+coût(s))/(1+CoûtTotal)} & \text{si} \quad \Delta Nav(s_0, s) \leq s_{NAV} \\ \frac{q_{NAV}(s)}{(1+coût(s))/(1+CoûtTotal)} * \sin(\frac{q_{NAV}(s)}{(2/\pi)*q_{NAV}(s)}) & \text{si} \quad \Delta Nav(s_0, s) > s_{NAV} \end{cases}$$

où

$$\Delta Nav(s_0, s) = |q_{NAV}(s) - q_{NAV}(s_0)|$$

La fonction $q_{NAV}(s)$ calcule le score de navigabilité de la solution s (cfr. 5.1).

$$Amé\_Nav = \frac{1 + \Delta Nav(s_0, s)}{2} \in [0, 1]$$

8.5 Comparaison des outils bayésiens

Cette section compare certains outils manipulant les réseaux bayésiens (inspirée de http ://www.cs.ubc.ca/∼murphyk/Bayes/bnsoft.html en Décembre 2008).

Les tableaux 8.1 et 8.2 listent les noms des outils (Noms), leurs auteurs (Auteurs), la possibilité d'avoir les sources et dans ce cas le langage de programmation associé (Sources), la présence d'une API (API), la présence d'une interface graphique (GUI), sa gratuité (Gratuit) et les méthodes d'inférences (Méthodes d'inférence).

Tableau 8.1 – Outils bayésiens (1ère partie)

Noms	**Auteurs**	**Sources**	**API**	**GUI**	**Gratuit**	**Méthodes d'inférence**
AgenaRisk	Agena	NON	OUI	OUI	Logiciel commercial	JTree
Analytica	Lumina	NON	OUI	OUI	Logiciel commercial	sampling
Banjo	Hartemink	Java	OUI	NON	Gratuit ou version académique	aucun
Bassist	U. Helsinki	C++	OUI	NON	Gratuit ou version académique	MH
Bayda	U. Helsinki	Java	OUI	OUI	Gratuit ou version académique	?
BayesBuilder	Nijman (U. Nijmegen)	NON	NON	OUI	Gratuit ou version académique	?
BayesiaLab	Bayesia Ltd	NON	NON	OUI	Logiciel commercial	jtree,G
Bayesware Discoverer	Bayesware	NON	NON	OUI	Logiciel commercial	?
B-course	U. Helsinki	NON	NON	OUI	Gratuit ou version académique	?
Belief net power constructor	Cheng (U.Alberta)	NON	NON	OUI	Gratuit ou version académique	?
BNT	Murphy (U.C.Berkeley)	Matlab/C	OUI	NON	Gratuit ou version académique	Many
BNJ	Hsu (Kansas)	Java	OUI	OUI	Gratuit ou version académique	jtree, IS
BucketElim	Rish (U.C.Irvine)	C++	OUI	NON	Gratuit ou version académique	Varelim
BUGS	MRC/Imperial College	NON	NON	NON	Gratuit ou version académique	Gibbs
Business Navigator 5	Data Digest Corp	NON	NON	OUI	Logiciel commercial	Jtree
CABeN	Cousins et al. (Wash. U.)	C	OUI	NON	Gratuit ou version académique	5 Sampling methods
Causal discoverer	Vanderbilt	NON	NON	-	Gratuit ou version académique	-
CoCo+Xlisp	Badsberg (U. Aalborg)	C/lisp	OUI	OUI	Gratuit ou version académique	Jtree
CIspace	Poole et al. (UBC)	Java	NON	OUI	Gratuit ou version académique	Varelim
DBNbox	Roberts et al	Matlab	-	NON	Gratuit ou version académique	Various
Deal	Bottcher et al	R	-	OUI	Gratuit ou version académique	None
DeriveIt	DeriveIt LLC	NON	-	-	Logiciel commercial	Jtree
Ergo	Noetic systems	NON	OUI	OUI	Logiciel commercial	jtree
GDAGsim	Wilkinson (U. Newcastle)	C	OUI	NON	Gratuit ou version académique	Exact
Genie	U. Pittsburgh	NON	NON	NON	Gratuit ou version académique	Jtree
GMRFsim	Rue (U. Trondheim)	C	OUI	NON	Gratuit ou version académique	MCMC

Tableau 8.2 – Outils bayésiens (2ème partie)

Noms	**Auteurs**	**Sources**	**API**	**GUI**	**Gratuit**	**Méthodes d'inférence**
GMTk	Bilmes (UW), Zweig (IBM)	NON	OUI	NON	Gratuit ou version académique	Jtree
gR	Lauritzen et al.	R	-	-	Gratuit ou version académique	-
Grappa	Green (Bristol)	R	-	NON	Gratuit ou version académique	Jtree
Hugin Expert	Hugin	NON	OUI	NON	Logiciel commercial	Jtree
Hydra	Warnes (U.Wash.)	Java	-	OUI	Gratuit ou version académique	MCMC
Ideal	Rockwell	Lisp	OUI	OUI	Gratuit ou version académique	Jtree
Java Bayes	Cozman (CMU)	Java	OUI	OUI	Gratuit ou version académique	Varelim, jtree
KBaseAI	Codeas	NON	OUI	NON	Logiciel commercial	varelim
LibB	Friedman (Hebrew U)	NON	OUI	NON	Gratuit ou version académique	aucun
MIM	HyperGraph Software	NON	NON	OUI	Logiciel commercial	Jtree
MSBNx	Microsoft	NON	OUI	NON	Gratuit ou version académique	Jtree
Netica	Norsys	NON	NON	NON	Logiciel commercial	jtree
Optimal Reinsertion	Moore, Wong (CMU)	NON	NON	NON	Gratuit ou version académique	aucun
PMT	Pavlovic (BU)	Matlab/C	-	NON	Gratuit ou version académique	special purpose
PNL	Eruhimov (Intel)	C++	-	NON	Gratuit ou version académique	Jtree
Pulcinella	IRIDIA	Lisp	OUI	OUI	Gratuit ou version académique	?
RISO	Dodier (U.Colorado)	Java	OUI	OUI	Gratuit ou version académique	Polytree
Sam Iam	Darwiche (UCLA)	NON	NON	OUI	Gratuit ou version académique	Recursive conditioning
Tetrad	CMU	NON	NON	NON	Gratuit ou version académique	None
UnBBayes	?	Java	-	OUI	Gratuit ou version académique	jtree
Vibes	Winn & Bishop (U. Cambridge)	Java	OUI	OUI	Gratuit ou version académique	Variational
Web Weaver	Xiang (U.Regina)	Java	OUI	OUI	Gratuit ou version académique	?
WinMine	Microsoft	NON	NON	OUI	Gratuit ou version académique	None
XBAIES 2.0	Cowell (City U.)	NON	NON	OUI	Gratuit ou version académique	Jtree